

**Oracle Database 11g:
入門 SQL 基礎 I Ed 1**

Volume 1

D49996JP10
Production 1.0
November 2007
D53411

ORACLE®

原著者

Puja Singh

© Oracle Corporation 2007 © 日本オラクル株式会社 2007

この文書には、米国Oracle Corporation及び日本オラクル株式会社が権利を有する情報が含まれており、使用と開示に対して定められたライセンス契約に従って提供されるものです。また、これらは著作権法による保護も受けています。ソフトウェアのリバース・エンジニアリングは禁止されています。

日本語版翻訳監修

郭 峰

この文書が合衆国政府の国防省関連機関に配布される場合は、次の制限付き権利が適用されます。

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

当社の事前の書面による承諾なしに、いかなる形式あるいはいかなる方法でも、本書及び本書に付属する資料の全体又は一部を複製することを禁じます。いかなる複製も著作権法違反であり、民事または刑事、もしくは両者の制裁の対象となります。

この文書が合衆国政府の国防総省以外の機関に配布される場合は、その権利は、FAR 52.227-14「一般データにおける権利」(Alternate IIIを含む)(June 1987)で定める権利の制限を受けます。

この文書の内容は予告なく変更されることがあります。

米国Oracle Corporation及び日本オラクル株式会社は、本書及び本書に付属する資料についてその記載内容に誤りがない事及び特定目的に対する適合性に関するいっさいの保証を行うものではありません。また、本書を参考にアプリケーション・ソフトウェアを作成された場合であっても、そのアプリケーション・ソフトウェアに関して米国Oracle Corporation及び日本オラクル株式会社(その関連会社も含みます)は一切の責任を負いかねます。

ORACLEは、Oracle Corporationおよびその関連会社の登録商標です。

本書で参照されているその他全ての製品やサービスの名称は、それぞれを表示する為に引用されており、それぞれ各社の商標である場合があります。

Oracle Internal & Oracle Academy
Use Only

目次

I はじめに	
章の目的	I- 2
章の講義項目	I- 3
コースの目的	I- 4
コースの講義項目	I- 5
このコースで使用する付録	I- 7
章の講義項目	I- 8
Oracle Database 11 ^g 対象分野	I- 9
Oracle Database 11 ^g	I-10
Oracle Fusion Middleware	I-12
Oracle Enterprise Manager Grid Control 10 ^g	I-13
Oracle BI Publisher	I-14
章の講義項目	I-15
リレーショナルおよびオブジェクト・リレーショナル・データベース管理システム	I-16
異なるメディアへのデータの格納	I-17
リレーショナル・データベースの概念	I-18
リレーショナル・データベースの定義	I-19
データ・モデル	I-20
E-Rモデル	I-21
エンティティ・リレーションシップモデリング規則	I-23
複数の表の関係付け	I-25
リレーショナル・データベースの用語	I-27
章の講義項目	I-29
SQLを使用したデータベースの問合せ	I-30
SQL文	I-31
SQLの開発環境	I-32
章の講義項目	I-33
Oracle SQL Developerの概要	I-34
Oracle SQL Developerの仕様	I-35
Oracle SQL Developerのインタフェース	I-36
データベース接続の作成	I-38
データベース・オブジェクトの参照	I-41
SQL Worksheetの使用方法	I-42
SQL文の実行	I-45
SQLコードの書式設定	I-46
SQL文の保存	I-47
スクリプト・ファイルの実行	I-48
Oracle SQL DeveloperからのSQL*Plusの起動	I-49

SQL*Plus内のSQL文	I-50
章の講義項目	I-51
Human Resources (HR)スキーマ	I-52
このコースで使用する表	I-53
章の講義項目	I-54
Oracle Database 11gのドキュメント	I-55
その他のリソース	I-56
まとめ	I-57
演習I: 概要	I-58
1 SQL SELECT文を使用したデータの検索	
目的	1- 2
章の講義項目	1- 3
SQL SELECT文の機能	1- 4
基本的なSELECT文	1- 5
すべての列の選択	1- 6
特定の列の選択	1- 7
SQL文の記述	1- 8
列ヘッダーのデフォルト設定	1- 9
章の講義項目	1-10
算術式	1-11
算術演算子の使用方法	1-12
演算子の優先順位	1-13
NULL値の定義	1-14
算術式のNULL値	1-15
章の講義項目	1-16
列別名の定義	1-17
列別名の使用方法	1-18
章の講義項目	1-19
連結演算子	1-20
リテラル文字列	1-21
リテラル文字列の使用方法	1-22
代替引用符(q)演算子	1-23
重複行	1-24
章の講義項目	1-25
表構造の表示	1-26
DESCRIBEコマンドの使用方法	1-27
まとめ	1-28
演習1: 概要	1-29

2 データの制限とソート

目的	2- 2
章の講義項目	2- 3
選択を使用した行の制限	2- 4
選択される行の制限	2- 5
WHERE句の使用方法	2- 6
文字列と日付	2- 7
比較演算子	2- 8
比較演算子の使用方法	2- 9
BETWEEN演算子を使用した範囲条件	2-10
IN演算子を使用したメンバーシップ条件	2-11
LIKE演算子を使用したパターン一致	2-12
ワイルドカード文字の組合せ	2-13
NULL条件の使用方法	2-14
論理演算子を使用した条件の定義	2-15
AND演算子の使用方法	2-16
OR演算子の使用方法	2-17
NOT演算子の使用方法	2-18
章の講義項目	2-19
優先順位の規則	2-20
章の講義項目	2-22
ORDER BY句の使用方法	2-23
ソート	2-24
章の講義項目	2-26
置換変数	2-27
シングルアンパサンド置換変数の使用方法	2-29
置換変数での文字値および日付値の使用	2-31
列名、式およびテキストの指定	2-32
ダブルアンパサンド置換変数の使用方法	2-33
章の講義項目	2-34
DEFINEコマンドの使用方法	2-35
VERIFYコマンドの使用方法	2-36
まとめ	2-37
演習2: 概要	2-38

3 単一行関数を使用した出力のカスタマイズ

目的	3- 2
章の講義項目	3- 3
SQL関数	3- 4
SQL関数の2つのタイプ	3- 5

単一行関数	3- 6
章の講義項目	3- 8
文字関数	3- 9
大文字/小文字変換関数	3-11
大文字/小文字変換関数の使用方法	3-12
文字操作関数	3-13
文字操作関数の使用方法	3-14
章の講義項目	3-15
数値関数	3-16
ROUND関数の使用方法	3-17
TRUNC関数の使用方法	3-18
MOD関数の使用方法	3-19
章の講義項目	3-20
日付の操作	3-21
RR日付書式	3-22
SYSDATE関数の使用方法	3-24
日付の算術演算	3-25
日付を使用した算術演算子の使用方法	3-26
章の講義項目	3-27
日付操作関数	3-28
日付関数の使用方法	3-29
日付を使用したROUND関数およびTRUNC関数の使用方法	3-30
まとめ	3-31
演習3: 概要	3-32
4 変換関数と条件式の使用方法	
目的	4- 2
章の講義項目	4- 3
変換関数	4- 4
暗黙的なデータ型変換	4- 5
明示的なデータ型変換	4- 7
章の講義項目	4-10
日付を使用したTO_CHAR関数の使用方法	4-11
日付書式モデルの要素	4-12
日付を使用したTO_CHAR関数の使用方法	4-16
数値を使用したTO_CHAR関数の使用方法	4-17
TO_NUMBERおよびTO_DATE関数の使用方法	4-20
RR日付書式を使用したTO_CHARおよびTO_DATE関数の使用方法	4-22
章の講義項目	4-23
関数のネスト	4-24

章の講義項目	4-26
汎用関数	4-27
NVL関数	4-28
NVL関数の使用方法	4-29
NVL2関数の使用方法	4-30
NULLIF関数の使用方法	4-31
COALESCE関数の使用方法	4-32
章の講義項目	4-35
条件式	4-36
CASE式	4-37
CASE式の使用法	4-38
DECODE関数	4-39
DECODE関数の使用法	4-40
まとめ	4-42
演習4: 概要	4-43
5 グループ関数を使用した集計データのレポート	
目的	5- 2
章の講義項目	5- 3
グループ関数の概要	5- 4
グループ関数のタイプ	5- 5
グループ関数: 構文	5- 6
AVGおよびSUM関数の使用法	5- 7
MINおよびMAX関数の使用法	5- 8
COUNT関数の使用法	5- 9
DISTINCTキーワードの使用法	5-10
グループ関数とNULL値	5-11
章の講義項目	5-12
データのグループの作成	5-13
データのグループの作成: GROUP BY句の構文	5-14
GROUP BY句の使用法	5-15
複数の列によるグループ化	5-17
複数の列に対するGROUP BY句の使用法	5-18
グループ関数を使用した無効な問合せ	5-19
グループの結果の制限	5-21
HAVING句を使用したグループの結果の制限	5-22
HAVING句の使用法	5-23
章の講義項目	5-25
グループ関数のネスト	5-26

まとめ	5-27
演習5: 概要	5-28
6 複数の表のデータの表示	
目的	6- 2
章の講義項目	6- 3
複数の表からのデータの取得	6- 4
結合のタイプ	6- 5
SQL:1999構文を使用した表の結合	6- 6
あいまいな列名の修飾	6- 7
章の講義項目	6- 8
自然結合の作成	6- 9
自然結合によるレコードの取得	6-10
USING句による結合の作成	6-11
列名の結合	6-12
USING句によるレコードの取得	6-13
USING句での表別名の使用方法	6-14
ON句による結合の作成	6-15
ON句によるレコードの取得	6-16
ON句による3方向結合の作成	6-17
結合への追加条件の適用	6-18
章の講義項目	6-19
表自体の結合	6-20
ON句による自己結合の作成	6-21
章の講義項目	6-22
非等価結合	6-23
非等価結合によるレコードの取得	6-24
章の講義項目	6-25
外部結合による直接一致しないレコードの取得	6-26
内部結合と外部結合	6-27
左側外部結合 (LEFT OUTER JOIN)	6-28
右側外部結合 (RIGHT OUTER JOIN)	6-29
完全外部結合 (FULL OUTER JOIN)	6-30
章の講義項目	6-31
デカルト積	6-32
デカルト積の生成	6-33
クロス結合の作成	6-34
まとめ	6-35
演習6: 概要	6-36

7 副問合せによる問合せの解決方法	
目的	7- 2
章の講義項目	7- 3
副問合せによる問題の解決方法	7- 4
副問合せの構文	7- 5
副問合せの使用方法	7- 6
副問合せの使用に関するガイドライン	7- 7
副問合せの種類	7- 8
章の講義項目	7- 9
単一行副問合せ	7-10
単一行副問合せの実行	7-11
副問合せにおけるグループ関数の使用方法	7-12
HAVING句での副問合せ	7-13
この文の間違いは何か	7-14
内側の問合せから行が戻されない	7-15
章の講義項目	7-16
複数行副問合せ	7-17
複数行副問合せでのANY演算子の使用方法	7-18
複数行副問合せでのALL演算子の使用方法	7-19
章の講義項目	7-20
副問合せにおけるNULL値	7-21
まとめ	7-23
演習7: 概要	7-24
8 集合演算子の使用方法	
目的	8- 2
章の講義項目	8- 3
集合演算子	8- 4
集合演算子のガイドライン	8- 5
Oracleサーバーと集合演算子	8- 6
章の講義項目	8- 7
この章で使用する表	8- 8
章の講義項目	8-12
UNION演算子	8-13
UNION演算子の使用方法	8-14
UNION ALL演算子	8-16
UNION ALL演算子の使用方法	8-17
章の講義項目	8-18
INTERSECT演算子	8-19
INTERSECT演算子の使用方法	8-20

章の講義項目	8-21
MINUS演算子	8-22
MINUS演算子の使用方法	8-23
章の講義項目	8-24
SELECT文の一致	8-25
SELECT文の一致: 例	8-26
章の講義項目	8-27
集合演算子でのORDER BY句の使用方法	8-28
まとめ	8-29
演習8: 概要	8-30
9 データの操作	
目的	9- 2
章の講義項目	9- 3
データ操作言語	9- 4
表への新しい行の追加	9- 5
INSERT文の構文	9- 6
新しい行の挿入	9- 7
NULL値を持つ行の挿入	9- 8
特殊な値の挿入	9- 9
特定の日付値および時刻値の挿入	9-10
スクリプトの作成	9-11
別の表からの行のコピー	9-12
章の講義項目	9-13
表内のデータの変更	9-14
UPDATE文の構文	9-15
表内の行の更新	9-16
副問合せによる2列の更新	9-17
別の表に基づく行の更新	9-18
章の講義項目	9-19
表からの行の削除	9-20
DELETE文	9-21
表からの行の削除	9-22
別の表に基づく行の削除	9-23
TRUNCATE文	9-24
章の講義項目	9-25
データベース・トランザクション	9-26
データベース・トランザクション: 開始と終了	9-27
COMMIT文およびROLLBACK文の利点	9-28
明示的なトランザクション制御文	9-29
マーカーへの変更のロールバック	9-30
暗黙的なトランザクション処理	9-31
COMMITまたはROLLBACK前のデータの状態	9-33
COMMIT後のデータの状態	9-34

データのコミット	9-35
ROLLBACK後のデータの状態	9-36
ROLLBACK後のデータの状態: 例	9-37
文レベル・ロールバック	9-38
章の講義項目	9-39
読取り一貫性	9-40
読取り一貫性の実装	9-41
章の講義項目	9-42
SELECT文のFOR UPDATE句	9-43
FOR UPDATE句: 例	9-44
まとめ	9-46
演習9: 概要	9-47
10 DDL文を使用した表の作成および管理	
目的	10- 2
章の講義項目	10- 3
データベース・オブジェクト	10- 4
ネーミング規則	10- 5
章の講義項目	10- 6
CREATE TABLE文	10- 7
別のユーザーの表の参照	10- 8
DEFAULTオプション	10- 9
表の作成	10-10
章の講義項目	10-11
データ型	10-12
日時データ型	10-14
章の講義項目	10-15
制約の設定	10-16
制約のガイドライン	10-17
制約の定義	10-18
NOT NULL制約	10-20
UNIQUE制約	10-21
PRIMARY KEY制約	10-23
FOREIGN KEY制約	10-24
FOREIGN KEY制約: キーワード	10-26
CHECK制約	10-27
CREATE TABLE: 例	10-28
制約違反	10-29
章の講義項目	10-31
副問合せを使用した表の作成	10-32

章の講義項目	10-34
ALTER TABLE文	10-35
読取り専用の表	10-36
章の講義項目	10-37
表の削除	10-38
まとめ	10-39
演習10: 概要	10-40
11 その他のスキーマ・オブジェクトの作成	
目的	11- 2
章の講義項目	11- 3
データベース・オブジェクト	11- 4
ビューの概要	11- 5
ビューの利点	11- 6
単一ビューと複合ビュー	11- 7
ビューの作成	11- 8
ビューからのデータの取得	11-11
ビューの変更	11-12
複合ビューの作成	11-13
ビューでDML操作を実行するためのルール	11-14
WITH CHECK OPTION句の使用方法	11-17
DML操作の拒否	11-18
ビューの削除	11-20
演習11: パート1の概要	11-21
章の講義項目	11-22
順序	11-23
CREATE SEQUENCE文: 構文	11-25
順序の作成	11-26
NEXTVALおよびCURRVAL疑似列	11-27
順序の使用方法	11-29
順序値のキャッシュ	11-30
順序の変更	11-31
順序変更のガイドライン	11-32
章の講義項目	11-33
索引	11-34
索引の作成方法	11-36
索引の作成	11-37
索引作成のガイドライン	11-38
索引の削除	11-39
章の講義項目	11-40

シノニム	11-41
オブジェクトのシノニムの作成	11-42
シノニムの作成および削除	11-43
まとめ	11-44
演習11: パート2の概要	11-45

付録A: 演習の解答

付録B: 表の説明

付録C: Oracle結合構文

付録D: SQL*Plusの使用

付録E: SQL Developer GUIを使用したDMLおよびDDL操作の実行

追加の演習

追加の演習: 解答

Oracle Internal & Oracle Academy
Use Only

Oracle Internal & Oracle Academy
Use Only

I

はじめに

ORACLE®

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

章の目的

この章を終えると、次のことができるようになります。

- このコースの目標の理解
- Oracle Database 11gの機能の列挙
- リレーショナル・データベースの理論的および物理的側面に関する説明
- OracleサーバーによるRDBMSおよびオブジェクト・リレーショナル・データベース管理システム(ORDBMS)の実装の説明
- このコースに使用できる開発環境の理解
- Oracle SQL Developerの主要機能の説明と使用
- このコースで使用するデータベースおよびスキーマの説明

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

この章では、リレーショナル・データベース管理システム(RDBMS)とオブジェクト・リレーショナル・データベース管理システム(ORDBMS)について学習します。また、SQL文の実行および書式設定やレポートの目的に使用される、開発環境としてのOracle SQL DeveloperとSQL*Plusについても紹介します。

章の講義項目

- コースの目的、講義項目およびコースで使用する付録
- Oracle Database 11*g*と関連製品の概要
- リレーショナル・データベース管理の概念と用語の概要
- SQLとその開発環境の紹介
- Oracle SQL Developerの概要
- Oracle Database 11*g*のドキュメントとその他のリソース

ORACLE

Copyright © 2007, Oracle. All rights reserved.

コースの目的

このコースを終えると、次のことができるようになります。

- Oracle Database 11gの主要コンポーネントの理解
- SELECT文による表からの行および列データの取得
- ソート済および制限付きデータのレポートの作成
- カスタマイズされたデータを生成および取得するためのSQL関数の使用
- 複数の表からデータを取得するための複雑な問合せの実行
- Oracle Database 11g内のデータを更新するためのデータ操作言語(DML)文の実行
- スキーマ・オブジェクトを作成して管理するためのデータ定義言語(DDL)文の実行

ORACLE

Copyright © 2007, Oracle. All rights reserved.

コースの目的

このコースでは、Oracle Database 11gデータベース・テクノロジーを紹介します。このクラスでは、リレーショナル・データベースと強力なSQLプログラミング言語の基本概念について学習します。また、このコースでは、1つの表および複数の表に対する問合せの記述、表内のデータの操作、データベース・オブジェクトの作成、メタデータの問合せを行うために必要なSQLスキルについて説明します。

コースの講義項目

- 1日目:

- はじめに
- SQL SELECT文を使用したデータの取得
- データの制限とソート
- 単一行関数を使用した出力のカスタマイズ
- 変換関数と条件式の使用方法

- 2日目:

- グループ関数を使用した集計データのレポート
- 複数の表のデータの表示
- 副問合せによる問合せの解決方法
- 集合演算子の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

コースの講義項目

- 3日目:

- データの操作
- DDL文を使用した表の作成および管理
- その他のスキーマ・オブジェクトの作成

ORACLE

Copyright © 2007, Oracle. All rights reserved.

このコースで使用する付録

- 付録A: 演習の解答
- 付録B: 表の説明
- 付録C: Oracle結合構文
- 付録D: SQL*Plusの使用方法
- 付録E: Oracle SQL Developer GUIを使用したDMLおよびDDL操作の実行
- 追加の演習
- 追加の演習の解答

ORACLE

Copyright © 2007, Oracle. All rights reserved.

章の講義項目

- コースの目的、コースの講義項目およびこのコースで使用する付録
- **Oracle Database 11gと関連製品の概要**
- リレーショナル・データベース管理の概念と用語の概要
- SQLとその開発環境の紹介
- Oracle SQL Developerの概要
- このコースで使用するHRスキーマと表
- Oracle Database 11gのドキュメントとその他のリソース

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Database 11g 対象分野



インフラストラクチャ・
グリッド

情報管理

アプリケーション
開発

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Database 11g 対象分野

Oracle Database 11gは、対象とする次の分野に渡って幅広い機能を提供します。

- **インフラストラクチャ・グリッド:** オラクルのインフラストラクチャ・グリッド・テクノロジーを使用することにより、低コストのサーバーとストレージのプーリングが実現され、管理性、高可用性およびパフォーマンスの点で最高品質のサービスを提供するシステムの構築が可能になります。Oracle Database 11gでは、グリッド・コンピューティングの利点が確立され、拡張されます。また、グリッド・コンピューティングの利点を最大限に引き出すだけでなく、Oracle Database 11gには、費用効率の高い制御された方法で、変更を管理するための独自の変更管理機能が用意されています。
- **情報管理:** Oracle Database 11gでは、コンテンツ管理、情報統合および情報ライフ・サイクル管理領域の既存の情報管理機能が拡張されます。オラクルは、Extensible Markup Language (XML)、テキスト、空間、マルチメディア、医療画像、セマンティック・テクノロジーなどの高度なデータ型のコンテンツ管理機能を提供します。
- **アプリケーション開発:** Oracle Database 11gは、PL/SQL、Java/JDBC、.NETとWindows、PHP、SQL Developer、Application Expressなどの主要なすべてのアプリケーション開発環境を使用し、管理できる機能を備えています。

Oracle Database 11g



管理性

高可用性

パフォーマンス

セキュリティ

情報の統合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Database 11g

組織は、昼夜を問わずにビジネス・アプリケーションへの高速で安全なアクセスを要求するユーザーに対して、数テラバイトもの情報提供をサポートする必要があります。データベース・システムは、信頼性が高く、どのような障害発生時でも迅速にリカバリできるものでなければなりません。

Oracle Database 11gは、組織がインフラストラクチャ・グリッドを簡単に管理して高品質サービスを提供する手助けとなるように、次の機能性を考慮して設計されています。

- **管理性:** 変更管理、管理自動化、障害診断機能のいくつかを使用することにより、データベース管理者 (DBA) は生産性の向上、コスト削減とエラーの最小化、およびサービスの最大限の品質向上を実現することができます。効果的な管理の促進に役立つ機能には、データベース・リプレイ機能、SQLパフォーマンス・アナライザ、自動SQLチューニング機能があります。
- **高可用性:** 高可用性機能を使用することにより、停止時間とデータ損失のリスクを軽減できます。これらの機能により、オンラインでの操作性が向上し、より高速なデータベース・アップグレードが可能になります。

Oracle Database 11g



管理性

高可用性

パフォーマンス

セキュリティ

情報統合

ORACLE

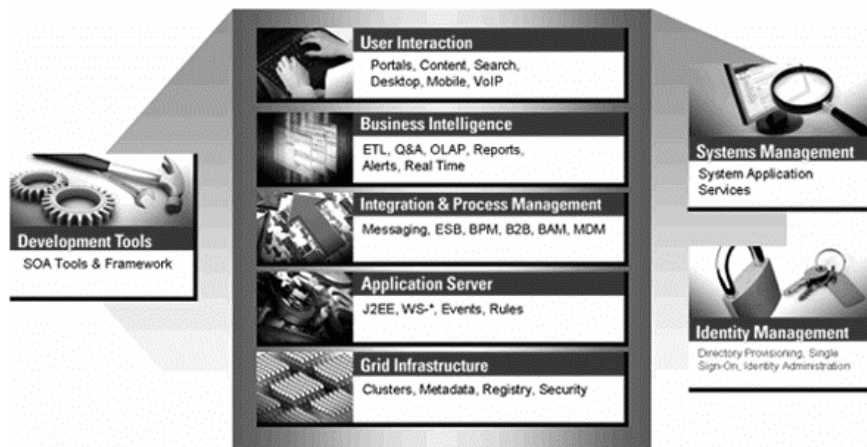
Copyright © 2007, Oracle. All rights reserved.

Oracle Database 11g(続き)

- **パフォーマンス:** SecureFile、オンライン・トランザクション処理 (OLTP) 用の圧縮、Real Application Clusters (RAC) の最適化、結果キャッシュなどの機能を使用することにより、データベースのパフォーマンスを大幅に向上させることができます。組織は、Oracle Database 11gを使用して、低コストのモジュール形式のストレージを使用して高速データ・アクセスを提供する、大規模なスケーラブル・トランザクション対応のデータ・ウェアハウス・システムを管理できます。
- **セキュリティ:** Oracle Database 11gを使用する組織は、一意のセキュアな構成、データ暗号化とマスキング、および高度な監査機能を使用して組織の情報を保護できます。Oracle Database 11gでは、すべてのタイプの情報に対して信頼性の高い高速アクセスを実現するために、業界標準のインタフェースを使用して、セキュアでスケーラブルなプラットフォームを提供しています。
- **情報統合:** Oracle Database 11gには、企業全体でデータを的確に統合するための様々な機能が用意されています。また、高度な情報ライフ・サイクル管理機能もサポートしています。この機能はデータベース内でのデータ変更の管理に役立ちます。

Oracle Fusion Middleware

顧客に定評のある、標準ベースの主要ソフトウェア製品のポートフォリオです。J2EEおよび開発者ツールから、統合サービス、ビジネス・インテリジェンス、コラボレーション、およびコンテンツ管理に至る様々なツールとサービスが含まれています。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Fusion Middleware

Oracle Fusion Middlewareは、サービス指向アーキテクチャ(SOA)の開発、配置および管理の完全なサポートを提供する、総合的で高度に統合された製品ファミリです。SOAは、簡単に統合して再使用できるモジュール形式のビジネス・サービスの開発を容易にすることにより、開発コストおよび保守コストの削減と、高品質のサービスの提供を実現します。Oracle Fusion Middlewareのプラグ可能なアーキテクチャでは、既存のアプリケーション、システム、またはテクノロジーへの投資を活用できます。その堅牢なコア・テクノロジーによって、計画内または計画外の停止による中断は最小限に抑えられます。

Oracle Fusion Middlewareファミリには、次の製品が含まれています。

- **エンタープライズ・アプリケーション・サーバー**: Application Server
- **統合およびプロセス管理**: BPEL Process Manager、Oracle Business Process Analysis Suite
- **開発ツール**: Oracle Application Development Framework、JDeveloper、SOA Suite
- **ビジネス・インテリジェンス**: Oracle Business Activity Monitoring、Oracle Data Integrator
- **システム管理**: Enterprise Manager
- **ID管理**: Oracle Identity Management
- **コンテンツ管理**: Oracle Content Database Suite
- **ユーザー相互作用**: Portal、WebCenter

Oracle Enterprise Manager Grid Control 10^g

- Oracle Fusion Middlewareによる効率的な管理
- アプリケーションおよびインフラストラクチャのライフ・サイクル管理の簡略化
- データベース管理およびアプリケーション管理機能の向上



ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Enterprise Manager Grid Control 10^g

Oracle Enterprise Manager Grid Control 10^gでは、OracleおよびOracle以外のシステムで、アプリケーション、ミドルウェアおよびデータベースの管理にわたる統合エンタープライズ管理が可能です。

Oracle Enterprise Manager Grid Control 10^gの特徴は、ビジネス・アプリケーションが依存するサービスに対する、SOA、Business Activity Monitoring、Identity Managementなどの高度なOracle Fusion Middleware管理機能です。

- サービスレベル管理、アプリケーションのパフォーマンス管理、構成管理、変更の自動化などのアプリケーションを対象とする、**広範囲にわたる管理機能**です。
- **組み込みのグリッド自動化機能**により、変動する需要に情報テクノロジーが積極的に対応し、業績向上につながる新しいサービスが迅速に実装されます。
- 特注アプリケーション、Oracle E-Business Suite、PeopleSoft、Siebel、Oracle Fusion Middleware、Oracle Database、基底インフラストラクチャなど、様々なアプリケーションを対象とする**徹底診断と、すぐに適用可能な処置**があります。
- **広範なライフ・サイクル管理機能**により、アプリケーションとインフラストラクチャのライフ・サイクル全体(運用上のテスト、ステージング、生産)に対応するソリューションを提供して、グリッド・コンピューティングを拡張します。同期化されたパッチ適用の同期、オペレーティング・システムの追加サポート、および競合検出機能を備えた簡略化されたパッチ管理があります。

Oracle BI Publisher

- セキュアな複数の形式で情報の作成、管理および配布を行うための集中アーキテクチャの提供
- あらゆる種類のレポートを開発、テストおよび配布する際の複雑さと時間の軽減
 - 財務レポート、請求書、販売または購入注文書、XML、EDI/EFT (eTextドキュメント)
- 柔軟なカスタマイズを実現
 - たとえば、Microsoft Wordドキュメントのレポートを、PDF、HTML、Excel、RTFなどの複数の形式で生成できる。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle BI Publisher

Oracle Database 11gには、Oracle BI Publisherも含まれています。これはオラクルが提供するエンタープライズ・レポート・ソリューションです。Oracle BI Publisher (以前はXML Publisherと呼ばれていた)は、複雑な分散環境で利用できる最も効率の良いスケーラブルなレポート・ソリューションを提供します。

Oracle BI Publisherを使用すると、ビジネス・ドキュメントの開発、カスタマイズおよび維持に関連する高コストが削減されるうえ、レポート管理の効率が向上します。ユーザーは、使い慣れた一連のデスクトップ・ツールを使用して、ITスタッフまたは開発者が作成したデータ問合せに基づいて独自のレポート形式を作成し、維持することができます。

Oracle BI Publisherのレポート形式は、ほとんどのユーザーがすでに習熟したツールであるMicrosoft WordまたはAdobe Acrobatを使用して設計できます。Oracle BI Publisherでは、複数のデータ・ソースから1つの出力ドキュメントにデータを書き出すこともできます。レポートを配布する場合は、プリンタ、電子メールまたはFAXを使用します。レポートはポータルに公開できます。Web-based Distributed Authoring and Versioning (WebDav) Webサーバー上でユーザーが共同でレポートを編集し管理できるようにすることも可能です。

章の講義項目

- コースの目的、コースの講義項目およびこのコースで使用する付録
- Oracle Database 11gと関連製品の概要
- リレーショナル・データベース管理の概念と用語の概要
- SQLとその開発環境の紹介
- Oracle SQL Developerの概要
- このコースで使用するHRスキーマと表
- Oracle Database 11gのドキュメントとその他のリソース

ORACLE

Copyright © 2007, Oracle. All rights reserved.

リレーショナルおよびオブジェクト・リレーショナル・データベース管理システム

- リレーショナル・モデルとオブジェクト・リレーショナル・モデル
- ユーザー定義データ型とオブジェクト
- リレーショナル・データベースとの完全な互換性
- マルチメディアおよびラージ・オブジェクトのサポート
- 高品質のデータベース・サーバー機能



ORACLE

Copyright © 2007, Oracle. All rights reserved.

リレーショナルおよびオブジェクト・リレーショナル・データベース管理システム

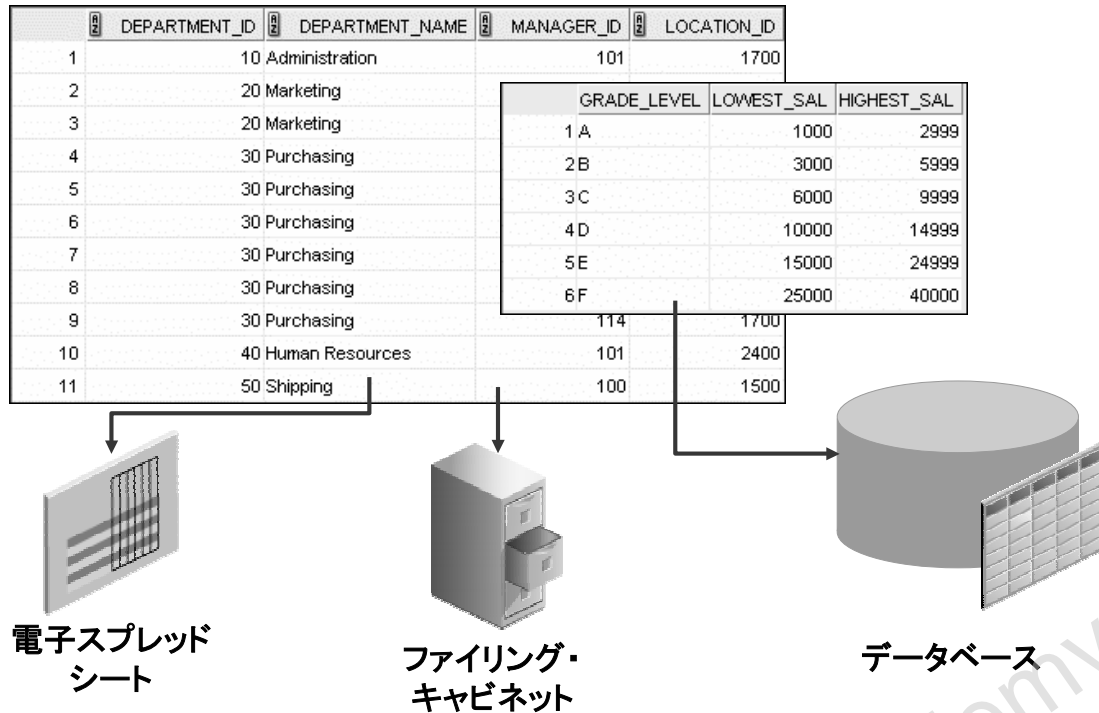
Oracleサーバーは、リレーショナル・データベース・モデルとオブジェクト・リレーショナル・データベース・モデルの両方をサポートします。

Oracleサーバーはデータ・モデリング機能を拡張して、オブジェクト指向プログラミング、複雑なデータ型、複雑なビジネス・オブジェクト、およびリレーショナル業界との完全な互換性を提供するオブジェクト・リレーショナル・データベース・モデルをサポートします。

ランタイム・データ構造の効果的な共有、バッファ・キャッシュの容量拡大、遅延可能制約など、OLTPアプリケーションのパフォーマンスと機能を向上させるための機能がいくつか含まれています。データ・ウェアハウス・アプリケーションは、挿入、更新および削除操作の平行実行、パーティション化、平行問合せの最適化などの機能拡張を利用できます。Network Computing Architecture (NCA) フレームワーク内で動作する場合、Oracleモデルは、分散され多重化されたクライアント/サーバーおよびWebベース・アプリケーションをサポートします。

リレーショナル・モデルおよびオブジェクト・リレーショナル・モデルの詳細は、『Oracle Database概要11gリリース1 (11.1)』のマニュアルを参照してください。

異なるメディアへのデータの格納



Copyright © 2007, Oracle. All rights reserved.

異なるメディアへのデータの格納

どの組織にもなんらかの情報ニーズがあります。図書館であれば、利用者、書籍、返却日および罰金のリストを維持します。企業であれば、従業員、部門および給与に関する情報を保存する必要があります。このような情報をデータと呼びます。

組織は、ファイリング・キャビネット内のハード・コピー・ドキュメント、電子スプレッドシートまたはデータベースに格納されたデータなど、様々なメディアに様々な形式でデータを保存できます。

データベースは、整理された情報のコレクションです。

データベースを管理するには、データベース管理システム(DBMS)が必要です。DBMSは、要求に応じてデータベースへのデータの格納、データベース内のデータの取得と変更を行うプログラムです。データベースには、階層、ネットワーク、リレーショナル、(最も新しい)オブジェクト・リレーショナルの4つの主要なタイプがあります。

リレーショナル・データベースの概念

- 1970年にDr. E. F. Coddがデータベース・システムのリレーショナル・モデルを提案しました。
- これがリレーショナル・データベース管理システム(RDBMS)の基礎です。
- リレーショナル・モデルは、次の要素で成り立ちます。
 - オブジェクトまたはリレーションのコレクション
 - リレーションに対して作用する一連の演算子
 - 精度と一貫性を保つためのデータ整合性

ORACLE

Copyright © 2007, Oracle. All rights reserved.

リレーショナル・データベースの概念

リレーショナル・モデルの原理は、1970年6月に「A Relational Model of Data for Large Shared Data Banks」という論文でDr. E. F. Coddによって初めて概要が示されました。この論文の中で、Dr. Coddはデータベース・システムのリレーショナル・モデルが提案されています。

当時使用されていた一般的なモデルは、階層モデルおよびネットワーク・モデルであり、単純なフラットファイル・データ構造である場合もありました。リレーショナル・データベース管理システム(RDBMS)は、特にその使いやすさと柔軟性に富んだ構造が評価されて急速に普及しました。さらに、オラクルなどの様々な革新的ベンダーが、強力なアプリケーション開発およびユーザー・インタフェース製品群でRDBMSを補完することにより、総合的なソリューションが提供されるようになりました。

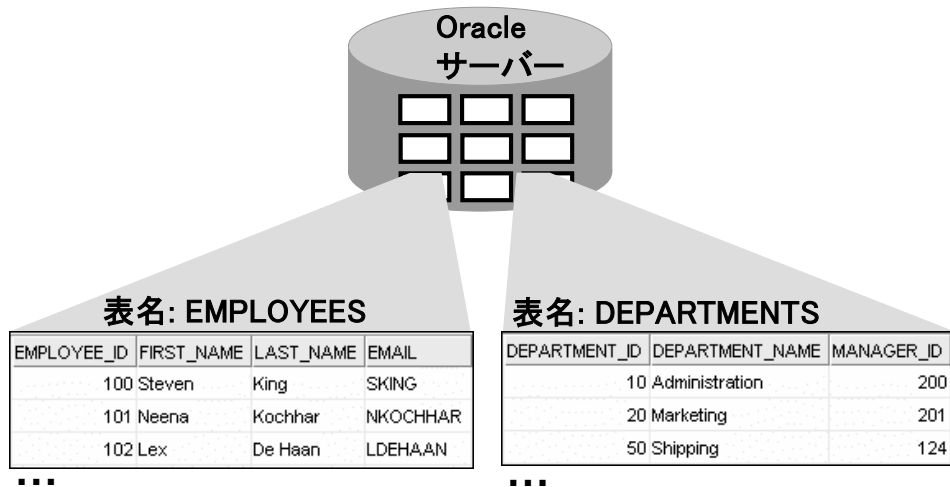
リレーショナル・モデルのコンポーネント

- データを格納するオブジェクトまたはリレーションのコレクション
- リレーションに作用して他のリレーションを生成できる一連の演算子
- 精度と一貫性を保つためのデータ整合性

詳細は、Chris Dateの著書『An Introduction to Database Systems, Eighth Edition』(Addison-Wesley、2004年)を参照してください。

リレーショナル・データベースの定義

リレーショナル・データベースは、リレーションまたは2次元表のコレクションです。



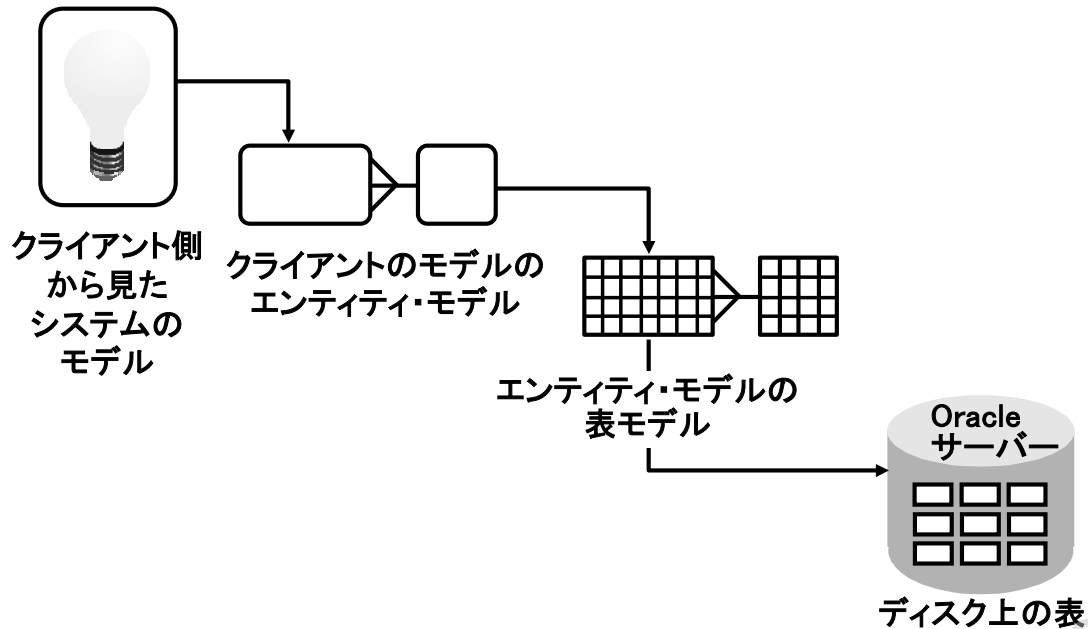
ORACLE

Copyright © 2007, Oracle. All rights reserved.

リレーショナル・データベースの定義

リレーショナル・データベースでは、リレーションまたは2次元表を使用して情報を格納します。たとえば、社内のすべての従業員に関する情報を格納する場合があります。リレーショナル・データベースでは、従業員表、部門表、給与表などの複数の表を作成して、従業員に関する各種の情報を格納します。

データ・モデル



Copyright © 2007, Oracle. All rights reserved.

データ・モデル

モデルは設計の土台となります。自動車のエンジニアの場合、モデルを構築して詳細を解決してから生産に移します。同じように、システム設計者は、モデルを開発してアイデアを模索し、データベース設計の理解を深めます。

モデルの目的

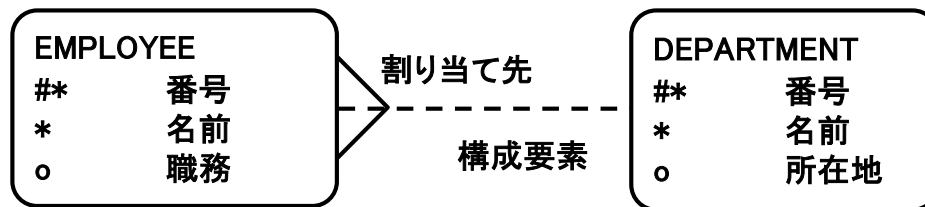
モデルは、頭の中の概念を伝える場合に役立ちます。次のような行為に使用できます。

- 伝達
- 分類
- 説明
- 指定
- 調査
- 発展
- 分析
- 模倣

目標は、このような多くの用途に適し、エンド・ユーザーが理解可能で、開発者向けの十分な詳細が定義されたモデルを作成して、データベース・システムを構築することです。

E-Rモデル

- 次のように業務仕様またはナレーティブからE-Rダイアグラムを作成します。



- シナリオ:
 - “... 1人以上の従業員を部門に割り当てる ...”
 - “... 一部の部門には、従業員が割り当てられていない...”

ORACLE

Copyright © 2007, Oracle. All rights reserved.

E-Rモデル

効果的なシステムでは、データを別個のカテゴリまたはエンティティに分類します。エンティティ・リレーションシップ (E-R) モデルとは、ビジネス内の様々なエンティティとそれらのリレーションシップの説明です。E-Rモデルは、業務仕様またはナレーティブから作成し、システム開発ライフ・サイクルの分析段階に構築します。E-Rモデルでは、ビジネスで要求される情報をビジネスで実行される活動から分離します。ビジネスの活動は変更される可能性はありますが、情報のタイプは一定である傾向があります。したがって、データ構造も一定である傾向があります。

E-Rモデル(続き)

E-Rモデリングの利点:

- 組織の情報を明確で正確な形式で文書化
- 情報が必要な範囲を明確に示す図を提供
- データベース設計に関するわかりやすいマップ図の提供
- 複数のアプリケーションを統合するための効果的フレームワークの提供

主要コンポーネント

- **エンティティ:** 知らせる必要がある情報に関する、重要な要素。例として部門、従業員、発注があります。
- **属性:** エンティティを説明または修飾するもの。たとえば、従業員エンティティの場合は、従業員番号、名前、職務、雇用日、部門番号などが属性になります。属性は、それぞれ必須または任意です。この状態を選択性と呼びます。
- **リレーションシップ:** 選択性と次数を示すエンティティ間の名前付き関連性。例として、従業員と部門、発注と品目があります。

Oracle Internal & Oracle Academy
Use Only

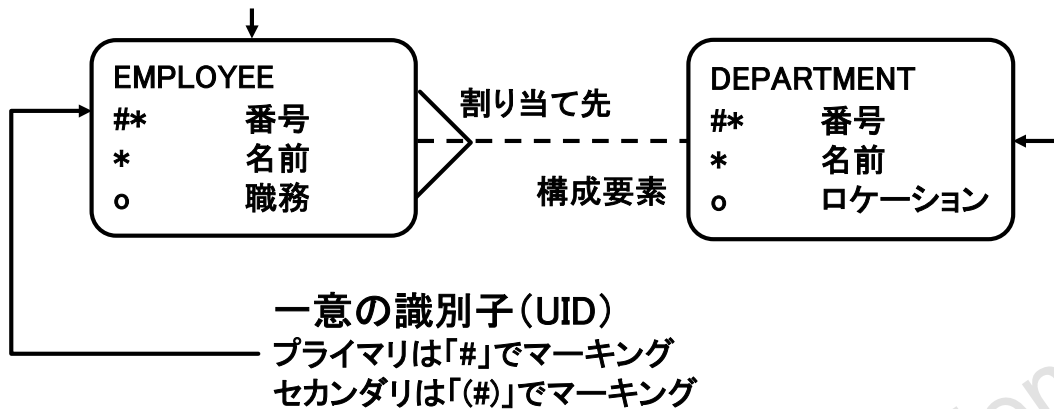
エンティティ・リレーションシップ モデリング規則

エンティティ:

- 単一の一意の名前を使用
- 大文字
- ソフト・ボックス
- シノニムはカッコ内

属性:

- 単一の名前を使用
- 小文字
- 必須は「*」でマーキング
- 任意は「o」でマーキング



ORACLE

Copyright © 2007, Oracle. All rights reserved.

E-Rモデリング規則

エンティティ

モデル内のエンティティを表すには、次の規則に従います。

- 単一の一意のエンティティ名を使用
- エンティティ名は大文字
- ソフト・ボックス
- 任意のシノニム名は、カッコ()内に大文字で指定

属性

モデル内の属性を表すには、次の規則に従います。

- 単一の小文字の名前を使用
- 必須属性(必ず知らせる値)にはアスタリスク「*」タグを使用
- 任意属性(任意で知らせる値)には文字「o」タグを使用

リレーションシップ

記号	説明
点線	任意であることを表す任意要素
実線	必須であることを表す必須要素
カラスの足のような形	1つ以上であることを表す次数要素
単一の線	1つだけであることを表す次数要素

E-Rモデリング規則(続き)

リレーションシップ

リレーションシップの各方向には次の要素が含まれます。

- **ラベル:** たとえば、「取得元」、「割り当て先」など
- **選択性:** 必須または任意
- **次数:** 1つのみ、または1つ以上

注意: 「カーディナリティ」という用語は「次数 (degree)」という用語のシノニムです。

それぞれのソース・エンティティは、{任意|必須}で宛先エンティティとのリレーションを {1つのみ|1つ以上}を持ちます。

注意: 規則は、時計回りに読み取ることです。

一意の識別子

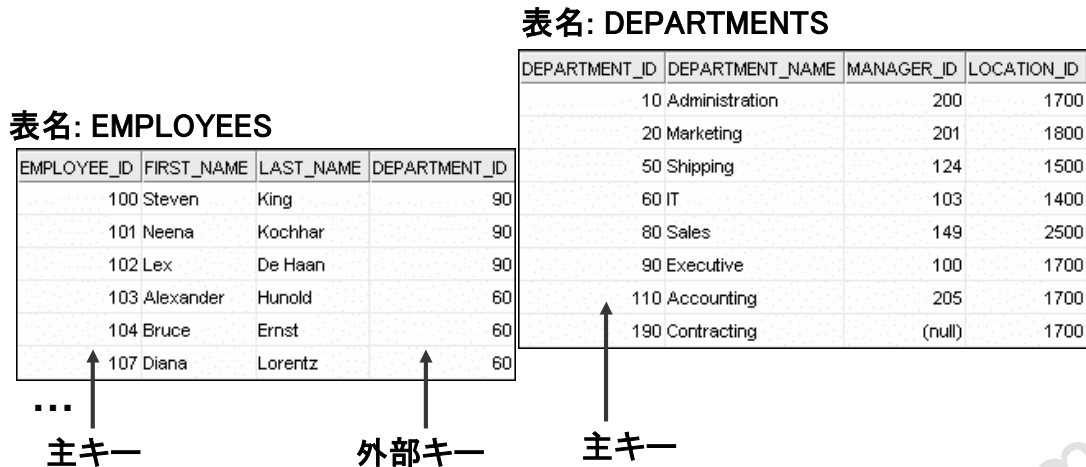
一意の識別子 (UID) は、属性またはリレーションシップ (あるいはその両方) の任意の組合せであり、エンティティの存在を識別する機能があります。それぞれのエンティティの存在は、一意に識別できるものでなければなりません。

- UIDの一部となるそれぞれの属性には、タグとしてハッシュ記号「#」を付けます。
- セカンダリUIDにはカッコで囲んだハッシュ記号 (#) を付けます。

Oracle Internal & Oracle Academy
Use Only

複数の表の関係付け

- 表内の各データ行は主キーによって一意に識別されます。
- 外部キーを使用して、複数の表のデータを論理的に関係付けることができます。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数の表の関係付け

それぞれの表には、厳密に1つのエンティティのみを説明するデータが含まれます。たとえば、EMPLOYEES表には、従業員に関する情報が含まれます。データのカテゴリが各表の先頭行にリストされ、その下に個々のケースが表示されます。表形式を使用すると、情報を簡単に視覚化して理解し、使用することができます。

異なるエンティティに関するデータは異なる表に格納されるので、特定の質問に答えるには、複数の表の結合が必要になる場合があります。たとえば、従業員が所属する部門の所在地を調べるとします。このシナリオでは、EMPLOYEES表(従業員に関するデータを含む)と

DEPARTMENTS 表(部門に関する情報を含む)から情報を取得する必要があります。RDBMSでは、1つの表内のデータを別の表内のデータに関係付ける場合に、外部キーを使用することができます。外部キーとは、同じ表または別の表内の主キーを参照する列(または列のセット)です。

1つの表内のデータを別の表内のデータに関係付ける機能を使用して、別の管理可能ユニットの情報を編成することができます。従業員データは、部門データと別の表に格納することで、論理的に区別された状態で保持できます。

複数の表の関係付け(続き)

主キーと外部キーのガイドライン

- 主キーには重複値を使用できません。
- 通常、主キーは変更できません。
- 外部キーはデータ値に基づく、単なる論理的な(物理的ではない)ポインタです。
- 外部キーの値は、既存の主キーの値または一意のキー値に一致する必要があります。一致しない場合は、NULLになる必要があります。
- 外部キーは、主キーまたは一意のキーの列を参照する必要があります。

Oracle Internal & Oracle Academy
Use Only

リレーショナル・データベースの用語

2	3	4	5	6		
1	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
	100	Steven	King	24000	(null)	90
	101	Neena	Kochhar	17000	(null)	90
	102	Lex	De Haan	17000	(null)	90
	103	Alexander	Hunold	9000	(null)	60
	104	Bruce	Ernst	6000	(null)	60
	107	Diana	Lorentz	4200	(null)	60
	124	Kevin	Mourgos	5800	(null)	50
	141	Trenna	Rajs	3500	(null)	50
	142	Curtis	Davies	3100	(null)	50
	143	Randall	Matos	2600	(null)	50
	144	Peter	Vargas	2500	(null)	50
	149	Eleni	Zlotkey	10500	0.2	80
	174	Ellen	Abel	11000	0.3	80
	176	Jonathon	Taylor	8600	0.2	80
	178	Kimberely	Grant	7000	0.15	(null)
	200	Jennifer	Whalen	4400	(null)	10
	201	Michael	Hartstein	13000	(null)	20
	202	Pat	Fay	6000	(null)	20
	205	Shelley	Higgins	12000	(null)	110
	206	William	Gietz	8300	(null)	110

Copyright © 2007, Oracle. All rights reserved.

リレーショナル・データベースの用語

リレーショナル・データベースには、1つまたは多くの表を含めることができます。表は、RDBMSの基本的な記憶域構造です。表には、従業員、請求書、顧客など、実世界の事柄に関して必要なすべてのデータが保持されます。

このスライドは、EMPLOYEES表またはリレーションの内容を示しています。番号が示す内容は次のとおりです。

1. 特定の従業員に関して必要なすべてのデータを表す1つの行（またはタプル）。表内の各行は、主キーで識別され、重複行は許可されません。行の順序は重要ではありません。データの取得時に、行の順序を指定します。
2. 従業員番号が含まれる列（または属性）。従業員番号は、EMPLOYEES表内の一意の従業員を識別します。この例では、従業員番号列が、主キーとして指定されています。主キーには、必ず値が含まれ、値は一意でなければなりません。
3. キー値ではない列。列は表内で1つの種類のデータを表します。この例の場合、このデータはすべての従業員の給与です。データ格納時の列の順序は重要ではありません。データの取得時に列の順序を指定します。

リレーショナル・データベースの用語(続き)

4. 部門番号が含まれている列で、外部キーでもあります。外部キーとは、表の相互の関係付けを定義する列です。外部キーは、同じ表または別の表内の主キーまたは一意のキーを参照します。この例では、DEPARTMENT_IDによってDEPARTMENTS表内の部門を一意に識別しています。
5. フィールドは、行と列の交差部分です。フィールド内には1つの値だけを含めることができます。
6. フィールドには値が含まれていない場合があります。これはNULL値と呼ばれます。EMPLOYEES表では、販売担当者のロールを持つ従業員のみがCOMMISSION_PCT(コミッション)フィールドに値が割り当てられています。

Oracle Internal & Oracle Academy
Use Only

章の講義項目

- コースの目的、コースの講義項目およびこのコースで使用する付録
- Oracle Database 11gと関連製品の概要
- リレーショナル・データベース管理の概念と用語の概要
- SQLとその開発環境の紹介
- Oracle SQL Developerの概要
- このコースで使用するHRスキーマと表
- Oracle Database 11gのドキュメントとその他のリソース

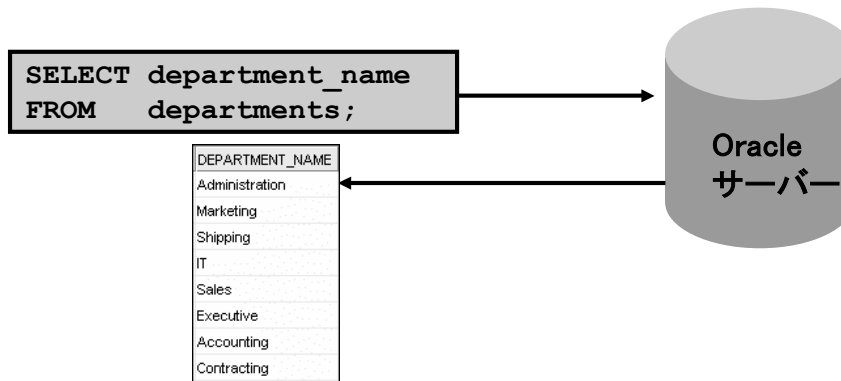
ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQLを使用したデータベースの問合せ

Structured Query Language (SQL) の特徴は次のとおりです。

- リレーショナル・データベースを操作するためのANSI標準言語
- 効率が良く、習得が容易で使いやすい機能
- 機能的な完全性 (SQLでは、表のデータの定義、取得および操作が可能)



ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQLを使用したデータベースの問合せ

リレーショナル・データベースでは、表へのアクセス経路を指定しません。また、データが物理的にどのように配置されているかを知る必要もありません。

データベースにアクセスするには、リレーショナル・データベースを操作するためのANSI (米国規格協会) 標準言語であるStructured Query Language (SQL) 文を実行します。SQLは、すべてのプログラムおよびユーザーがOracle Database内のデータにアクセスする際に使用する文のセットです。多くの場合、アプリケーション・プログラムとOracleツールでは、ユーザーはSQLを直接使用しなくてもデータベースにアクセスできます。ただし、これらのアプリケーションは、このユーザーのリクエストを実行するときにSQLを使用する必要があります。

SQLには、次のような様々なタスクを実行するための文が用意されています。

- データの問合せ
- 表内での行の挿入、更新および削除
- オブジェクトの作成、置換、変更および削除
- データベースおよびそのオブジェクトへのアクセスの制御
- データベースの一貫性と整合性の保証

SQLでは、これらのすべてのタスクを1つの一貫した言語で統一し、論理レベルでのデータの操作を実行できます。

SQL文

SELECT INSERT UPDATE DELETE MERGE	データ操作言語 (DML)
CREATE ALTER DROP RENAME TRUNCATE COMMENT	データ定義言語 (DDL)
GRANT REVOKE	データ制御言語 (DCL)
COMMIT ROLLBACK SAVEPOINT	トランザクション制御

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL文

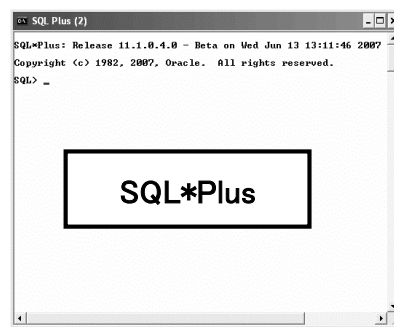
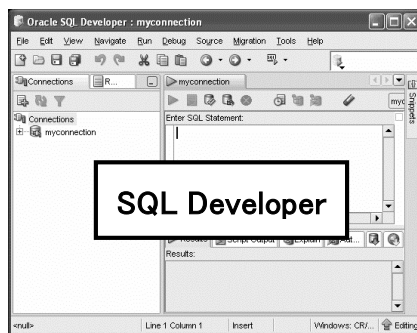
オラクルがサポートするSQL文は、業界標準に準拠しています。オラクル社では、主要担当者がSQL標準化委員会と積極的にかかわりを持ち、進化を続ける標準に今後も準拠していきます。業界で支持されている委員会は、ANSIとISO(国際標準化機構)です。ANSIとISOは、いずれもSQLをリレーショナル・データベースの標準言語として承認しています。

文	説明
SELECT INSERT UPDATE DELETE MERGE	それぞれ、データベースからのデータの取得、新しい行の入力、既存の行の変更、およびデータベースの不要な行の削除を実行します。これらを総称して、データ操作言語 (DML) と呼びます。
CREATE ALTER DROP RENAME TRUNCATE COMMENT	表のデータ構造の設定、変更および削除を実行します。これらを総称して、データ定義言語 (DDL) と呼びます。
GRANT REVOKE	Oracle Databaseとその内部構造に対するアクセス権を付与または削除します。
COMMIT ROLLBACK SAVEPOINT	DML文によって行われた変更を管理します。データへの変更は、まとめて論理トランザクションにグループ化できます。

SQLの開発環境

このコースでは、次の製品を使用します。

- 主に、Oracle SQL Developerリリース1.2を使用します。
- 次の場合は、SQL*Plusを使用します。
 - Oracle SQL Developerへのアクセス権がない場合
 - または、Oracle SQL Developerでコマンドが動作しない場合



ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQLの開発環境

このコースは、スライドと演習の例で説明されているSQL文を実行するためのツールとしてOracle SQL Developerを使用して開発されています。Oracle SQL Developerによってサポートされていないコマンドの場合は、SQL*Plus環境を使用してください。

章の講義項目

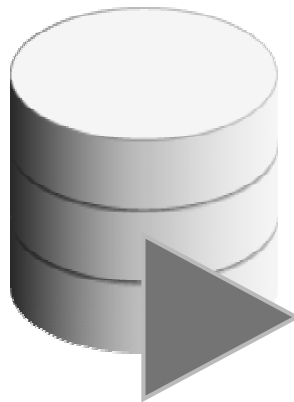
- コースの目的、コースの講義項目およびこのコースで使用する付録
- Oracle Database 11gと関連製品の概要
- リレーショナル・データベース管理の概念と用語の概要
- SQLとその開発環境の紹介
- **Oracle SQL Developerの概要**
- このコースで使用するHRスキーマと表
- Oracle Database 11gのドキュメントとその他のリソース

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle SQL Developerの概要

- Oracle SQL Developerは、生産性を向上させ、データベース開発タスクを簡略化するグラフィカル・ツールです。
- 標準のOracleデータベース認証を使用して、任意のターゲットOracle Databaseスキーマに接続できます。



SQL Developer

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle SQL Developerの概要

Oracle SQL Developerは、生産性の向上と、日常的なデータベース・タスクの開発の簡略化を目的として設計された無償のグラフィカル・ツールです。数回クリックするだけで、簡単にストアド・プロシージャの作成とデバッグ、SQL文のテスト、およびオブティマイザ計画の表示を実行できます。データベース開発用ビジュアル・ツールであるOracle SQL Developerにより、次のタスクが簡略化されます。

- データベース・オブジェクトの参照および管理
- SQL文およびスクリプトの実行
- PL/SQL文の編集およびデバッグ
- レポートの作成

標準のOracle Database認証を使用して、任意のターゲットOracle Databaseスキーマに接続できます。接続後は、データベース内のオブジェクトを操作できます。

注意: Oracle SQL Developerリリース1.2は、Developer Migration Workbenchと密接に統合されるため、移行リリースと呼ばれています。この統合によって、ユーザーは、サードパーティ・データベース内のデータベース・オブジェクトおよびデータの参照、およびこれらのデータベースからOracleへの移行をシングル・ポイントで行うことができます。MySQL、Microsoft SQL Server、Microsoft Accessなどの選択したサードパーティ(Oracle以外)データベース用のスキーマに接続し、これらのデータベース内のメタデータおよびデータを表示することもできます。

また、Oracle SQL Developerリリース1.2では、Oracle Application Expressリリース3.0.1 (Oracle APEX)もサポートされます。

Oracle SQL Developerの仕様

- Javaで開発されています。
- Windows、LinuxおよびMac OS Xプラットフォームをサポートします。
- JDBC Thinドライバを使用してデフォルトの接続が確立されます。
- インストーラは必要ありません。
 - ダウンロードしたOracle SQL Developerキットを解凍し、`sqldeveloper.exe`をダブルクリックしてOracle SQL Developerを起動する。
- リリース9.2.0.1以降のOracle Databaseに接続します。
- 次のリンクから無償でダウンロードできます。
 - http://www.oracle.com/technology/products/database/sql_developer/index.html
- JDK 1.5をシステムにインストールする必要があります。
次のリンクからダウンロードできます。
 - http://java.sun.com/javase/downloads/index_jdk5.jsp

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle SQL Developerの仕様

Oracle SQL Developerは、Oracle JDeveloper統合開発環境 (IDE) を活用してJavaで開発されたクロス・プラットフォーム・ツールです。Windows、LinuxおよびMacオペレーティング・システム (OS) X プラットフォーム上で動作します。Oracle SQL Developerをデータベース・サーバーにインストールして、デスクトップからリモートで接続できるので、クライアント/サーバーのネットワーク通信量が解消されます。

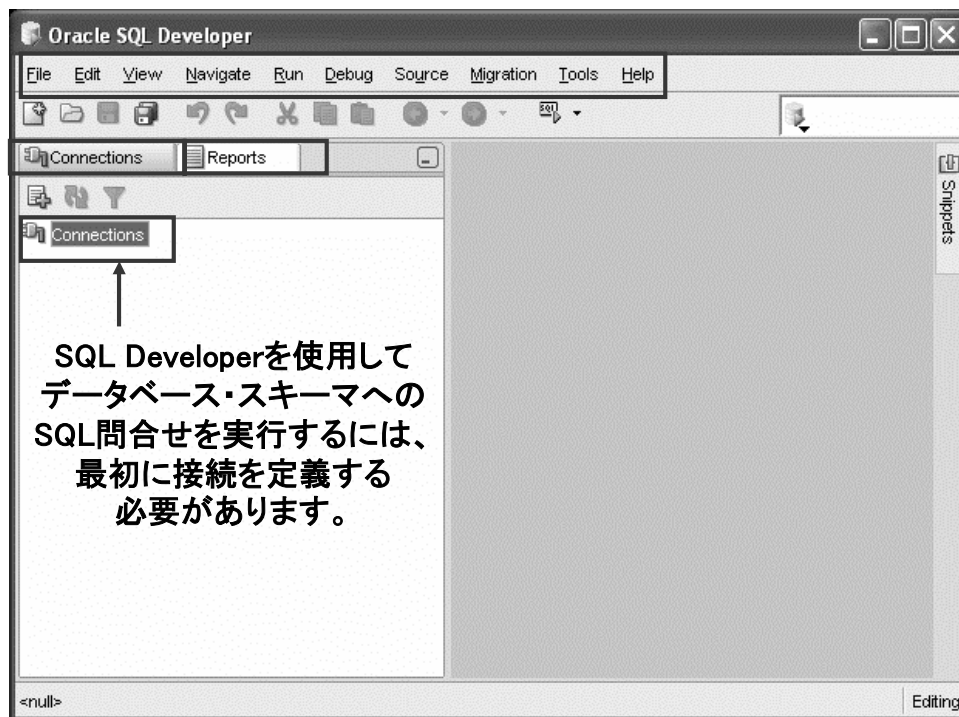
データベースへのデフォルト接続は、Java Database Connectivity (JDBC) Thinドライバを使用して確立されます。したがって、Oracleホームは必要ありません。Oracle SQL Developerはインストーラを必要としません。必要な作業は、ダウンロードしたファイルを解凍するだけです。Oracle SQL Developerでは、リリース9.2.0.1以降のOracle Databaseと、Express Editionを含めたすべてのエディションのOracle Databaseに接続できます。

Oracle SQL Developerは、次のパッケージ・オプションとともにダウンロードできます。

- Oracle SQL Developer for Windows (ダウンロード時にJava Development Kit (JDK) 1.5と一緒にダウンロードするかどうかを選択するオプション)
- Oracle SQL Developer for Multiple Platforms (JDK 1.5をインストール済みであることが必須)
- Oracle SQL Developer for Mac OS X プラットフォーム (JDK 1.5をインストール済みであることが必須)
- Oracle SQL Developer RPM for Linux (JDK 1.5をインストール済みであることが必須)

なお、Oracle SQL Developerリリース1.2はJDK 6.0での動作が保証されています。

Oracle SQL Developerのインタフェース



Copyright © 2007, Oracle. All rights reserved.

Oracle SQL Developerのインタフェース

Oracle SQL Developerには、次の2つの主要なナビゲーション・タブがあります。

- **「Connections」タブ:** このタブでは、アクセスが可能なデータベース・オブジェクトおよびユーザーを参照できます。
- **「Reports」タブ:** このタブでは、事前定義されたレポートの実行、または独自のレポートの作成と追加を行うことができます。

Oracle SQL Developerでは、ナビゲーション用の左ペインを使用してオブジェクトの検索と選択を行います。右ペインには、選択したオブジェクトに関する情報が表示されます。プリファレンスを設定して、Oracle SQL Developerの外観および動作の様々な要素をカスタマイズできます。上部のメニューには、標準のメニュー項目のほか、Oracle SQL Developer固有の機能のメニュー項目が含まれています。

1. **「View」:** Oracle SQL Developerインタフェースの表示内容に影響を与えるオプションが含まれています。
2. **「Navigate」:** ペインに移動したりサブプログラムの実行時に使用するオプションが含まれています。
3. **「Run」:** 関数またはプロシージャの選択時に必要になる「Run File」オプションと「Execution Profile」オプションが含まれています。
4. **「Debug」:** デバッグ用に関数またはプロシージャを選択するときに必要になるオプションが含まれています。
5. **「Source」:** 関数およびプロシージャの編集時に使用するオプションが含まれています。

Oracle SQL Developerのインタフェース (続き)

6. 「Migration」: サードパーティ・データベースをOracleに移行するときに必要になるオプションが含まれています。

7. 「Tools」: SQL*Plus、「Preferences」、「SQL Worksheet」などのツールを呼び出します。

注意: データベース・スキーマに接続して、SQL問合せの実行またはプロシージャや関数の実行を可能にするには、少なくとも1つの接続を定義する必要があります。

Oracle Internal & Oracle Academy
Use Only

データベース接続の作成

- Oracle SQL Developerを使用するには、1つ以上のデータベース接続が必要です。
- 次のものへの接続を作成してテストできます。
 - 複数のデータベース
 - 複数のスキーマ
- Oracle SQL Developerでは、システム上のtnsnames.oraファイルで定義されている接続が自動的にインポートされます。
- XMLファイルに接続をエクスポートできます。
- 作成した追加のデータベース接続は、それぞれ接続ナビゲータの階層に表示されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

データベース接続の作成

接続は、特定のデータベースにそのデータベースの特定ユーザーとして接続するために必要な情報を指定するOracle SQL Developerオブジェクトです。Oracle SQL Developerを使用するには、1つ以上のデータベース接続が必要です。既存の接続、作成した接続またはインポートした接続を使用できます。

複数のデータベースおよび複数のスキーマに対する接続を作成してテストできます。

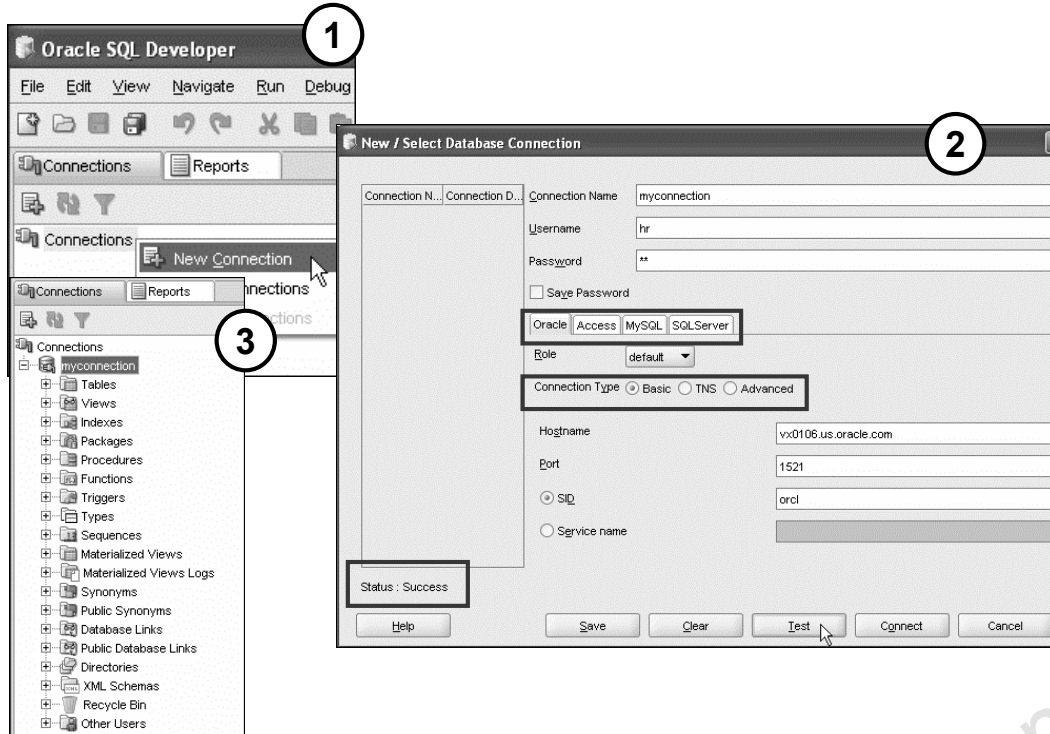
デフォルトでは、tnsnames.oraファイルは\$ORACLE_HOME/network/adminディレクトリに格納されます。ただし、TNS_ADMIN環境変数またはレジストリ値によって指定されたディレクトリに格納することもできます。Oracle SQL Developerを起動して、「Database Connections」ダイアログ・ボックスを表示すると、Oracle SQL Developerによって、システム上のtnsnames.oraファイルで定義されている接続が自動的にインポートされます。

注意: Windowsシステムでは、tnsnames.oraファイルが存在していても、Oracle SQL Developerでその接続を使用していない場合は、TNS_ADMINをシステム環境変数として定義してください。

後から再使用できるように、XMLファイルに接続をエクスポートできます。

異なるユーザーとして同じデータベースへの追加接続を作成するか、異なるデータベースに接続できます。

データベース接続の作成



Copyright © 2007, Oracle. All rights reserved.

データベース接続の作成(続き)

データベース接続を作成するには、次の手順を実行します。

1. 「Connections」タブ・ページで「Connections」を右クリックして、「New Connection」を選択します。
2. 「New/Select Database Connection」ウィンドウで、接続名を入力します。接続先のスキーマのユーザー名とパスワードを入力します。
 1. 「Role」ドロップダウン・リストから、「default」または「SYSDBA」を選択できます (sysユーザーまたはDBA権限を持つユーザーの場合は、「SYSDBA」を選択します)。
 2. 次の接続タイプを選択できます。
 - 「Basic」: このタイプでは、接続先のデータベースのホスト名とシステム識別子 (SID) を入力します。「Port」はすでに1521に設定されています。または、リモート・データベース接続を使用する場合は、直接「Service name」を入力することもできます。
 - 「TNS」: tnsnames.oraファイルからインポートされたデータベース別名のいずれかを選択します。
 - 「Advanced」: カスタムJDBC URLを定義して、データベースに接続します。
3. 「Test」をクリックして、接続が正しく設定されていることを確認します。
4. 「Connect」をクリックします。

「Save Password」チェック・ボックスを選択した場合は、パスワードがXMLファイルに保存されます。したがって、Oracle SQL Developer接続を閉じた後に再度開くと、パスワードを入力するように指示するメッセージは表示されません。

データベース接続の作成(続き)

3. 接続が接続ナビゲータに追加されます。その接続を展開すると、データベース・オブジェクトを表示し、依存関係、詳細、統計などのオブジェクト定義を表示できます。

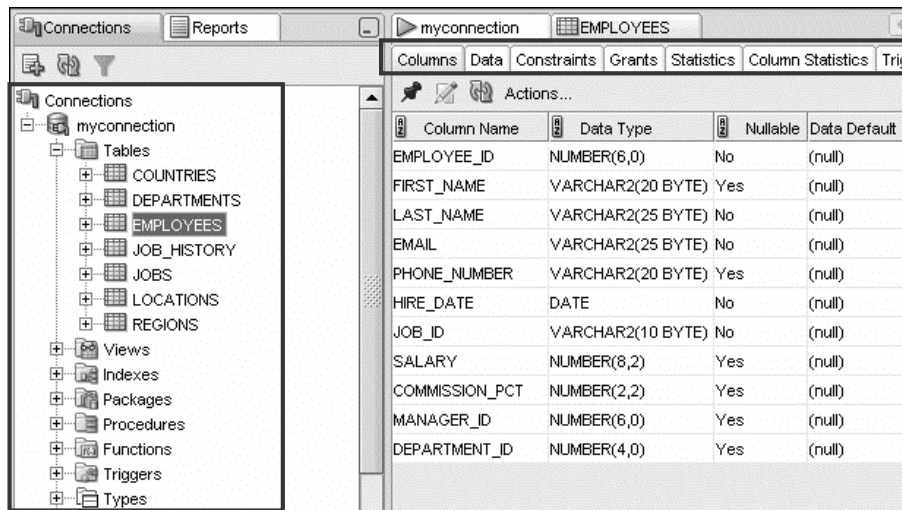
注意: 同じ「New/Select Database Connection」ウィンドウの「Access」、「MySQL」および「SQL Server」タブを使用して、Oracle以外のデータ・ソースへの接続を定義できます。ただし、これらの接続は、そのデータ・ソース内のオブジェクトおよびデータを参照可能にする読取り専用接続です。

Oracle Internal & Oracle Academy
Use Only

データベース・オブジェクトの参照

接続ナビゲータを使用すると、次のことを実行できます。

- データベース・スキーマ内の様々なオブジェクトを参照できる
- オブジェクトの定義を一目で確認できる



Copyright © 2007, Oracle. All rights reserved.

データベース・オブジェクトの参照

データベース接続を作成すると、接続ナビゲータを使用して、データベース・スキーマ内の表、ビュー、索引、パッケージ、プロシージャ、トリガー、タイプなどの様々なオブジェクトを参照できます。

Oracle SQL Developerでは、ナビゲーション用の左ペインでオブジェクトの検索と選択を行います。右ペインには、選択したオブジェクトに関する情報が表示されます。プリファレンスを設定して、Oracle SQL Developerの様々な外観をカスタマイズできます。

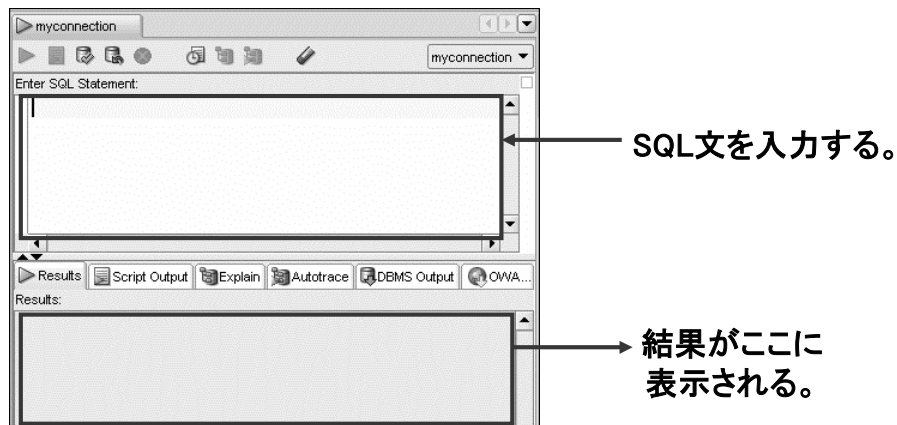
データ・ディクショナリから取り出された情報のタブに分類されているオブジェクトの定義を表示できます。たとえば、ナビゲータで表を選択すると、列、制約、権限の付与、統計、トリガーなどに関する詳細情報が読みやすい形式でタブ・ページに表示されます。

スライドに示されているEMPLOYEES表の定義を表示する場合は、次の手順を実行します。

1. 接続ナビゲータで「Connections」ノードを展開します。
2. 「Tables」を展開します。
3. 「EMPLOYEES」をクリックします。デフォルトでは、「Columns」タブが選択されています。このタブには、表の列の説明が表示されます。「Data」タブを使用すると、表のデータを表示できます。また、新しい行の入力、データの更新、これらの変更のデータベースへのコミットを行うこともできます。

SQL Worksheetの使用方法

- SQL Worksheetを使用して、SQL、PL/SQLおよびSQL*Plus文を入力し、実行します。
- ワークシートに関連付けられたデータベース接続によって処理できるアクションを指定します。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

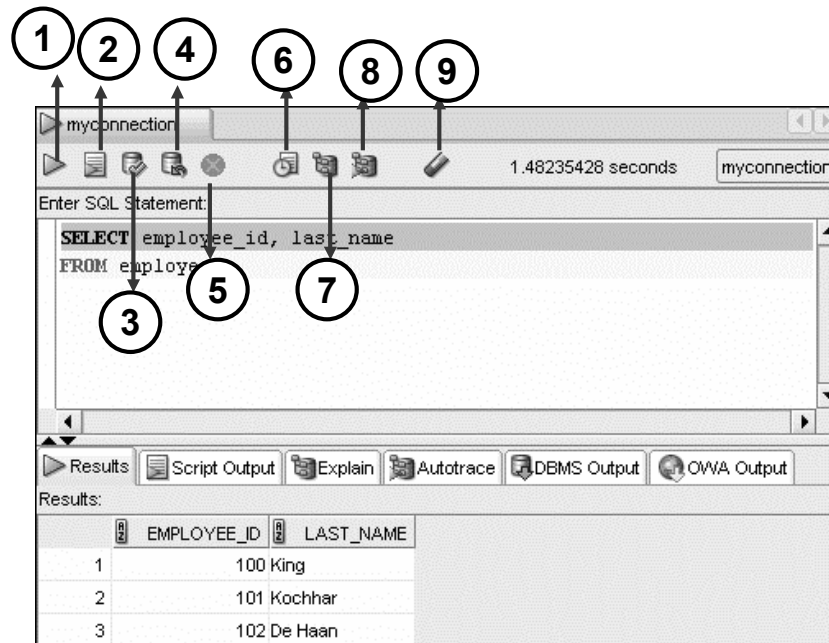
SQL Worksheetの使用方法

データベースに接続すると、その接続用のSQL Worksheetのウィンドウが自動的に開きます。このSQL Worksheetを使用すると、SQL、PL/SQLおよびSQL*Plus文を入力し、実行することができます。SQLおよびPL/SQLコマンドはSQL WorksheetからOracle Databaseに直接渡されるため、これらのコマンドはすべてサポートされます。ただし、Oracle SQL Developerで使用されるSQL*Plusコマンドは、データベースに渡される前にSQL Worksheetで解析する必要があります。SQL Worksheetでは、現在多くのSQL*Plusコマンドをサポートしています。SQL Worksheetによってサポートされないコマンドは無視され、Oracle Databaseに送信されません。SQL Worksheetを使用して、SQL文と一部のSQL*Plusコマンドを実行できます。

次の2つのオプションのいずれかを使用して、SQL Worksheetを表示できます。

- 「Tools」→「SQL Worksheet」を選択します。
- メイン・ツールバー上にある「Open SQL Worksheet」アイコンをクリックします。

SQL Worksheetの使用方法



ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL Worksheetの使用方法(続き)

ショートカット・キーまたはアイコンを使用して、SQL文の実行、スクリプトの実行、実行したSQL文の履歴の表示などの特定のタスクを行うこともできます。

SQL Worksheetツールバーのアイコンを使用すると、次のタスクを実行できます。

1. **「Execute Statement」**: 「Enter SQL Statement」ボックス内のカーソル位置の文が実行されます。また、[F9]を押す方法でも実行できます。通常、出力は「Results」タブ・ページに書式設定されて表示されます。
2. **「Run Script」**: Script Runnerを使用して「Enter SQL Statement」ボックス内のすべての文が実行されます。通常、出力は「Scripts」タブ・ページに従来のスクリプト形式で表示されます。
3. **「Commit」**: データベースへの変更が書き込まれ、トランザクションが終了します。
4. **「Rollback」**: データベースへの変更はデータベースに書き込まずに破棄され、トランザクションが終了します。
5. **「Cancel」**: 現在実行されている文の実行が停止されます。
6. **「SQL History」**: ダイアログ・ボックスに、実行したSQL文に関する情報が表示されます。
7. **「Execute Explain Plan」**: 実行計画が生成されます。「Explain」タブをクリックすると表示できます。

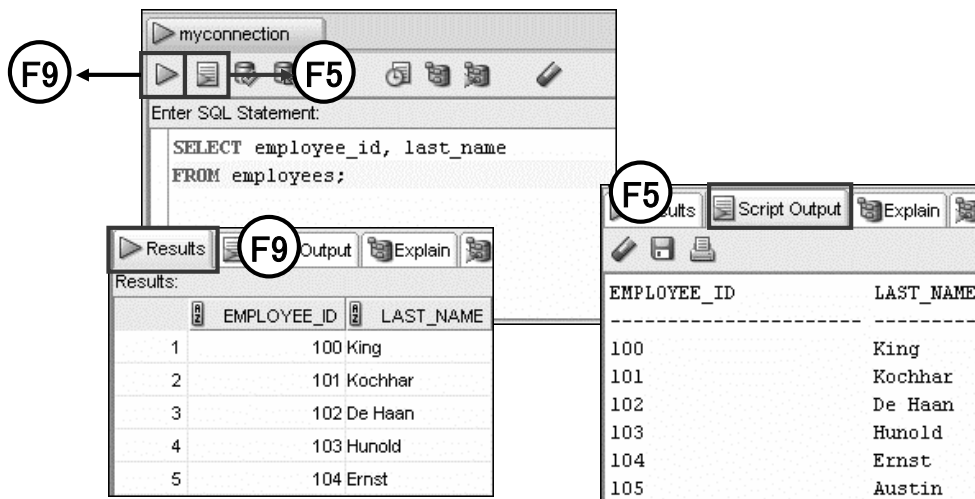
SQL Worksheetの使用方法(続き)

8. 「**Autotrace**」: 「Autotrace」アイコンをクリックしてSQL文を実行すると、トレース関連情報が表示されます。この情報はチューニングによってパフォーマンスが向上するSQL文を識別するのに役立つ場合があります。
9. 「**Clear**」: 「Enter SQL Statement」ボックス内の文が消去されます。[Ctrl]キーを押しながら[D]を押すことでも、文が消去されます。

Oracle Internal & Oracle Academy
Use Only

SQL文の実行

「Enter SQL Statement」ボックスに、1つまたは複数のSQL文を入力します。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL文の実行

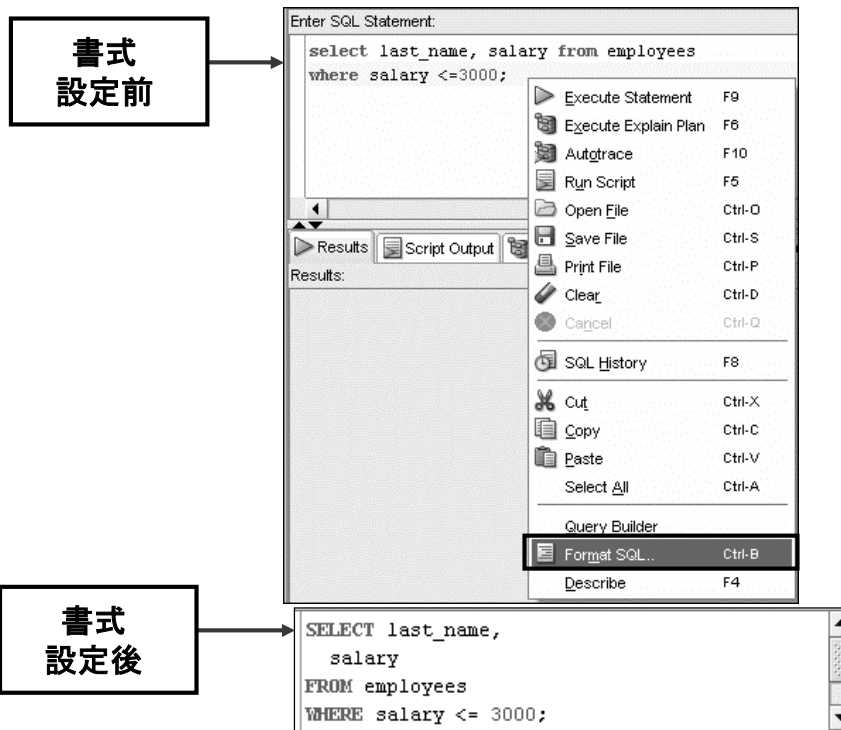
SQL Worksheetの「Enter SQL Statement」ボックスに、1つまたは複数のSQL文を入力できます。文が1つである場合、文末にセミコロンを付けるか付けないかは任意です。

文を入力すると、SQLキーワードが自動的にハイライトされます。SQL文を実行するには、カーソル位置がその文にあることを確認してから、「Execute Statement」アイコンをクリックします。[F9]を押す方法でも実行できます。

複数のSQL文を実行して結果を表示するには、「Run Script」アイコンをクリックします。[F5]を押す方法でも実行できます。

スライド内の例は、同じ問合せに関して、[F9]キーまたは「Execute Statement」を使用した場合と、[F5]または「Run Script」を使用した場合の出力の違いを示しています。

SQLコードの書式設定



ORACLE

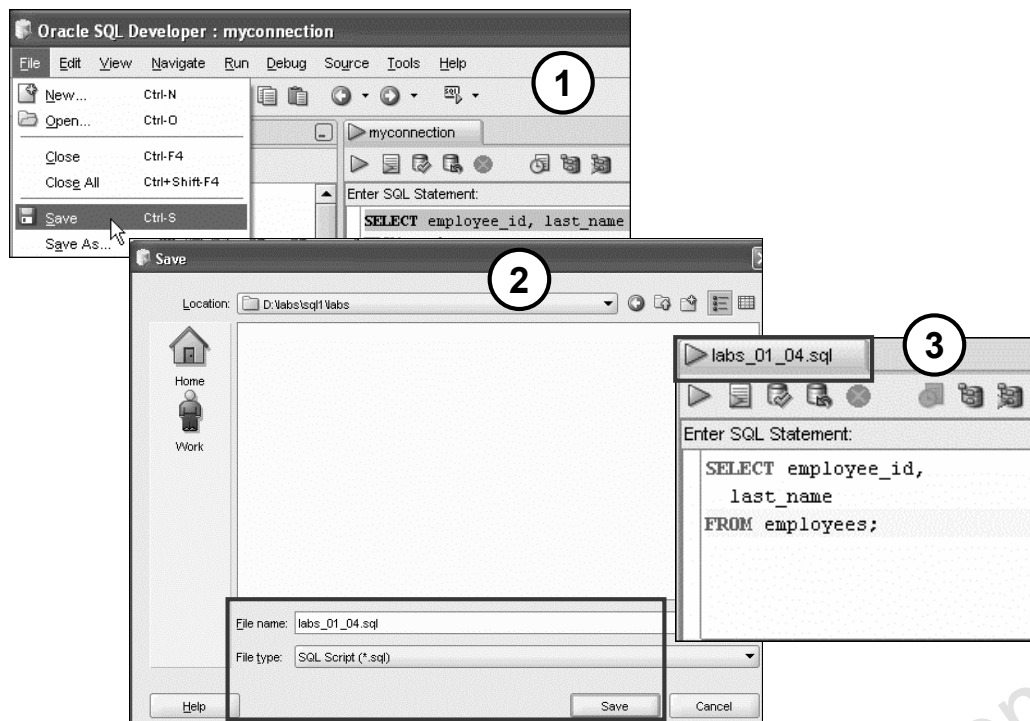
Copyright © 2007, Oracle. All rights reserved.

SQLコードの書式設定

SQLコードのインデント、間隔、大文字化、行分離などを行ってその外観を整えることもできます。Oracle SQL Developerには、SQLコードの書式を設定する機能があります。

SQLコードの書式を設定するには、文の領域内を右クリックして、「Format SQL」を選択します。スライドの書式設定前の例では、SQLコードのキーワードが小文字になっていて、文が適切にインデントされていません。書式設定後は、キーワードが大文字化されて、文が正しくインデントされることで、SQLコードの外観が整います。

SQL文の保存



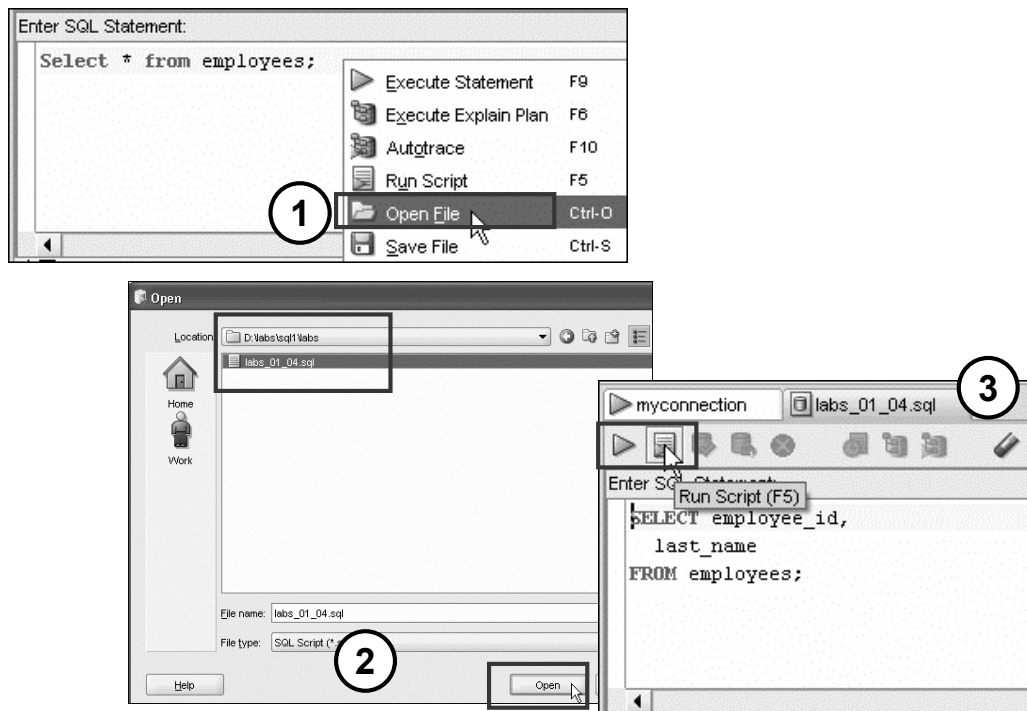
Copyright © 2007, Oracle. All rights reserved.

SQL文の保存

今後実施するほとんどの演習で、SQL Worksheet内の特定の問合せを.sqlファイルとして保存する必要があります。それには、次の手順を実行します。

1. 「File」メニューで「Save」または「Save As」を選択するか（現行の.sqlスクリプトの名前を変更する場合）、またはSQL Worksheetを右クリックして、「Save File」を選択します。[Ctrl]キーを押しながら[S]を押す方法でも実行できます。
2. 「Save」ダイアログ・ボックスで、適切なファイル名を入力します。拡張子が.sqlになっているか、「File type」として「SQL Script (*.sql)」が選択されていることを確認します。「Save」をクリックします。
注意: このコースでは、sqlスクリプトをD:\labs\sql\labsフォルダに保存する必要があります。
3. SQL Worksheetの名前が、スクリプトの保存時に指定したファイル名に変更されます。同じワークシートで他のSQL文を入力しないようにしてください。他のSQL問合せを続行するには、新しいワークシートを開きます。

スクリプト・ファイルの実行



Copyright © 2007, Oracle. All rights reserved.

スクリプト・ファイルの実行

保存した.sqlスクリプト・ファイルを実行するには、次の手順を実行します。

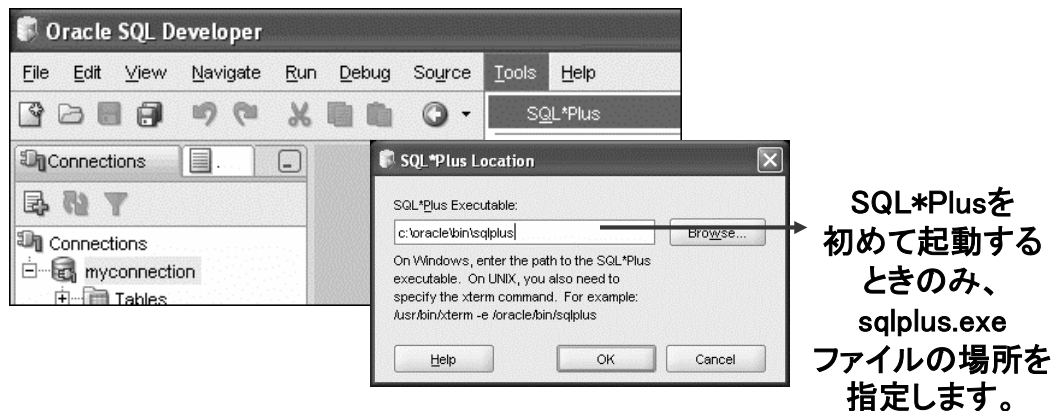
1. SQL Worksheetを右クリックして「Open File」を選択するか、「File」メニューから「Open」を選択します。[Ctrl]キーを押しながら[O]を押す方法でも実行できます。
2. 「Open」ダイアログ・ボックスで、D:\labs\sql\labsフォルダ、またはスクリプト・ファイルを保存した場所へ移動し、ファイルを選択して「Open」をクリックします。
3. スクリプト・ファイルが新しいワークシートで開きます。ここで、「Execute Statement」アイコンまたは「Run Script」アイコンをクリックして、スクリプトを実行できます。この場合も、同じワークシートで他のSQL文を入力しないようにしてください。他のSQL問合せの入力を続行する場合は、新しいワークシートを開きます。

注意: デフォルト・ディレクトリをD:\labs\sql\labsに設定して、スクリプトを開くかまたは保存しようとするたびに、SQL Developerがスクリプト検索用に同じパスを選択するようにできます。それには、「Tools」メニューで「Preferences」を選択します。「Preferences」ダイアログ・ボックスで「Database」を展開し、「Worksheet Parameters」を選択します。右ペインで「Browse」をクリックして、スクリプト検索用のデフォルト・パスを設定し、「OK」をクリックします。

注意: 他のデータ・オブジェクト作成およびデータ取得タスクにOracle SQL Developer GUIインタフェースを使用する方法については、付録E「Oracle SQL Developer GUIを使用したDMLおよびDDL操作の実行」を参照してください。

Oracle SQL Developerからの SQL*Plusの起動

Oracle SQL DeveloperからSQL*Plusコマンドライン・インタフェースを起動することができます。



Copyright © 2007, Oracle. All rights reserved.

Oracle SQL DeveloperからのSQL*Plusの起動

SQL WorksheetはほとんどのSQL*Plus文をサポートしています。SQL*Plus文は、SQL Worksheetで解析してからデータベースに渡す必要があります。SQL WorksheetによってサポートされていないSQL*Plus文は無視され、データベースに渡されません。「SQL*Plus」コマンド・ウィンドウを表示するには、「Tools」メニューから「SQL*Plus」を選択します。この機能を使用するには、Oracle SQL Developerを使用するシステムに、SQL*Plus実行可能ファイルが含まれているOracleホーム・ディレクトリまたはフォルダが存在する必要があります。SQL*Plus実行可能ファイルの場所がOracle SQL Developerプリファレンスにまだ格納されていない場合は、その場所を指定するように求められます。

注意:「Tools」→「SQL*Plus」メニュー・オプションが無効になっている場合は、接続ナビゲータでmyconnectionなどのデータベース接続をクリックします。このメニュー・オプションは、いずれかのSQL Worksheetがアクティブになっていると無効になります。

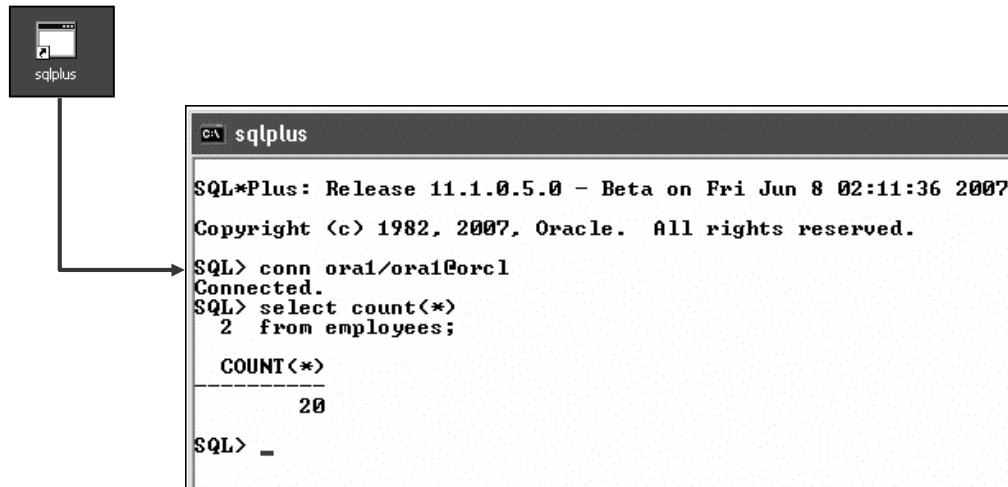
SQL WorksheetでサポートされないSQL*Plus文の例を、いくつか次に示します。

- append
- archive
- attribute
- break

SQL WorksheetによってサポートされているSQL*Plus文とサポートされていないSQL*Plus文の全リストについては、Oracle SQL Developerオンライン・ヘルプで、SQL WorksheetでサポートされるSQL*Plus文、サポートされないSQL*Plus文に関する項を参照してください。

SQL*Plus内のSQL文

Oracle Database 11gでのSQL*Plusは、コマンドライン・インタフェースです。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL*Plus内のSQL文

Oracle SQL*Plusは、実行するSQL文とPL/SQLブロックを送信し、アプリケーションまたはコマンド・ウィンドウに結果を受け取ることができるコマンドライン・インタフェースです。

SQL*Plusは次の形式で提供されます。

- データベースに付属
- クライアントおよびデータベース・サーバー・システム上にインストール済み
- アイコンまたはコマンドラインを使用してアクセス

注意: Oracle SQL Developerへのアクセス権がない場合にSQL*Plusを使用する必要があるときは、研修クラスのセットアップ時にデスクトップに「SQL*Plus」アイコンが作成されています。このアイコンは、Oracle SQL DeveloperがSQL*Plusコマンドをサポートしていない場合にも役立つことがあります。

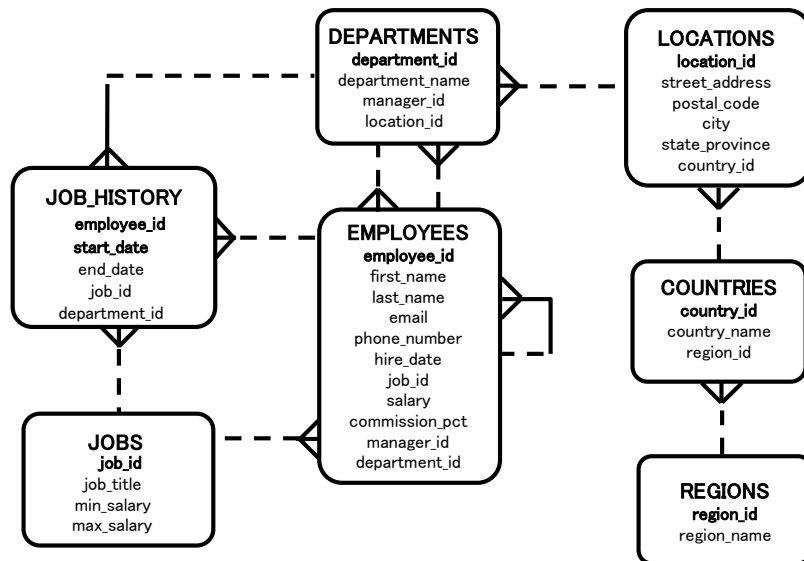
章の講義項目

- コースの目的、コースの講義項目およびこのコースで使用する付録
- Oracle Database 11gと関連製品の概要
- リレーショナル・データベース管理の概念と用語の概要
- SQLとその開発環境の紹介
- Oracle SQL Developerの概要
- このコースで使用するHRスキーマと表
- Oracle Database 11gのドキュメントとその他のリソース

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Human Resources (HR)スキーマ



ORACLE

Copyright © 2007, Oracle. All rights reserved.

Human Resources (HR)スキーマの説明

Human Resources (HR)スキーマは、Oracle DatabaseにインストールできるOracle Sample Schemasに含まれています。このコースの演習セッションでは、HRスキーマのデータを使用します。

表の説明

- REGIONSには、アメリカ、アジアなど、地域を表す行が含まれています。
- COUNTRIESには、国の行が含まれていて、それぞれに地域が関連付けられています。
- LOCATIONSには、特定の国内の企業の具体的な事業所、倉庫、生産拠点の住所が含まれています。
- DEPARTMENTSは、従業員が勤務する部門に関する詳細を示しています。それぞれの部門には、EMPLOYEES表内の部門マネージャを表すリレーションシップが含まれている場合があります。
- EMPLOYEESには、部門で勤務している各従業員に関する詳細が含まれています。一部の従業員については、どの部門にも割り当てられていない場合があります。
- JOBSには、それぞれの従業員が従事する可能性のある職種が含まれています。
- JOB_HISTORYには、従業員の職務歴が含まれています。職務の範囲内で従業員の部門が変更されたり、部門内で職務が変更されたりすると、従業員の以前の職務情報が含まれているこの表に新しい行が挿入されます。

このコースで使用する表

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Ellen	Abel	2900	(null)	50	RMATOS	650.121.2874	15-MAR-98

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	(null)	1700

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

DEPARTMENTS

JOB_GRADES

ORACLE

Copyright © 2007, Oracle. All rights reserved.

このコースで使用する表

このコースで使用する主な表は、次のとおりです。

- EMPLOYEES表: すべての従業員の詳細を示します。
- DEPARTMENTS表: すべての部門の詳細を示します。
- JOB_GRADES表: 様々な等級の給与の詳細を示します。

これらの表以外に、前のスライドで示したLOCATIONSやJOB_HISTORYなどの他の表も使用します。

注意: すべての表の構造とデータについては、付録Bで説明しています。

章の講義項目

- コースの目的、コースの講義項目およびこのコースで使用する付録
- Oracle Database 11gと関連製品の概要
- リレーショナル・データベース管理の概念と用語の概要
- SQLとその開発環境の紹介
- Oracle SQL Developerの概要
- このコースで使用するHRスキーマと表
- Oracle Database 11gのドキュメントとその他のリソース

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Database 11gのドキュメント

- 『Oracle Database新機能ガイド11gリリース1(11.1)』
- 『Oracle Databaseリファレンス11gリリース1(11.1)』
- 『Oracle Database SQL言語リファレンス11gリリース1(11.1)』
- 『Oracle Database概要11gリリース1(11.1)』
- 『Oracle Database SQL Developerユーザーズ・ガイド』

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Database 11gのドキュメント

Oracle Database 11gドキュメント・ライブラリを参照するには、
<http://www.oracle.com/pls/db111/homepage>にアクセスしてください。

その他のリソース

Oracle Database 11gに関する追加情報については、次のリソースを参照してください。

- 『Oracle Database 11g: New Features eStudies (英語)』
- 『Oracle by Example series (OBE): Oracle Database 11g (英語)』
 - http://www.oracle.com/technology/obe/11gr1_db/index.htm

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

- Oracle Database 11gは、次の利点または機能を拡張します。
 - インフラストラクチャ・グリッドの利点
 - 既存の情報管理機能
 - PL/SQL、Java/JDBC、.NET、XMLなどの主要なアプリケーションの開発環境を使用する機能
- データベースはORDBMSに基づいています。
- リレーショナル・データベースは、リレーションで構成され、関係演算によって管理され、データ整合性制約によって制御されます。
- Oracleサーバーでは、SQLを使用して情報の格納および管理を実行できます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

リレーショナル・データベース管理システムは、オブジェクトまたはリレーションで構成されます。オブジェクトやリレーションは、演算によって管理され、データ整合性制約によって制御されます。オラクル社は、ユーザーのRDBMSニーズに沿った製品およびサービスを提供しています。主要な製品は次のとおりです。

- Oracle Database 11g。SQLを使用して情報の格納および管理を実行します。
- Oracle Fusion Middleware。統合および再利用できるモジュール形式のビジネス・サービスを開発、配置および管理します。
- Oracle Enterprise Manager Grid Control 10g。グリッド環境内の一連のシステムをまたいで管理タスクを管理および自動化するために使用します。

SQL

Oracleサーバーでは、ANSI標準のSQLがサポートされ、拡張機能が含まれています。SQLは、サーバーと通信してデータのアクセス、操作および制御を行うための言語です。

演習I: 概要

この演習では次の項目について説明しています。

- Oracle SQL Developerのデモンストレーションの実行
- Oracle SQL Developerの起動、新しいデータベース接続の作成、およびHR表の参照

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習I: 概要

この演習では、次のことを実行します。

- Oracle SQL Developerのデモンストレーション全体を実行します。
- Oracle SQL Developerを使用して、自分に割り当てられたORAアカウントのデータ・オブジェクトを調べます。ORAアカウントには、HRスキーマ表が含まれています。

演習ファイルに関しては、次の場所を使用します。

D:\labs\SQL1\labs

演習ファイルを保存するように指示された場合は、この場所にファイルを保存してください。

演習によっては、実習のはじめに、“時間があるときは”、または“さらに演習を続ける場合は”という表現が使用されている場合があります。このような実習については、割り当てられた時間内に他のすべての実習が完了し、さらに高度なスキルにチャレンジする場合にのみ取り組んでください。

演習はゆっくり正確に実行してください。コマンド・ファイルの保存および実行を試すことができます。質問があるときは、いつでも講師に尋ねてください。

注意: すべての記述された演習問題は、開発環境としてOracle SQL Developerを使用します。Oracle SQL Developerを使用することをお勧めしますが、このコースで利用可能である、SQL*Plusを使用することもできます。

演習I

これは、このコースの様々な演習の中の最初の演習です。解答(必要な場合)は付録Aに記載されています。演習は、対応する章のほとんどのトピックを網羅するように考えられています。

Oracle SQL Developerのデモンストレーション全体の実行: データベース接続の作成

1. 次のサイトにあるデモンストレーション「Creating a database connection」にアクセスしてください。

http://st-curriculum.oracle.com/tutorial/SQLDeveloper/html/module2/mod02_cp_newdbconn.htm

Oracle SQL Developerの起動

2. 「sqldeveloper」デスクトップ・アイコンを使用して、Oracle SQL Developerを起動します。

注意: 初めてSQL Developerを起動するときは、java.exeファイルのパスを指定する必要があります。これはすでに研修クラスのセットアップの一部として行われています。どのような場合も、入力するように指示されたら次のパスを入力してください。

D:\app\Administrator\product\11.1.0\client_1\jdevstudio\jdk\bin

新しいOracle SQL Developerデータベース接続の作成

3. 新しいデータベース接続を作成するには、接続ナビゲータで「Connections」を右クリックします。メニューから「New Connection」を選択します。「New/Select Database Connection」ダイアログ・ボックスが表示されます。
4. 次の情報を使用して、データベース接続を作成します。
 - a. 「Connection Name」: myconnection.
 - b. 「Username」: oraxx。xxは、使用するPCの番号です。ora1～ora20の範囲のアカウントから、oraアカウントを1つ割り当てるよう講師に依頼してください。
 - c. 「Password」: oraxx
 - d. 「Hostname」: データベース・サーバーが稼働しているマシンのホスト名を入力します。
 - e. 「Port」: 1521
 - f. 「SID」: ORCL
 - g. 必ず「Save Password」チェック・ボックスを選択します。

演習I(続き)

Oracle SQL Developerデータベース接続を使用したテストと接続

5. 新しい接続をテストします。
6. 状態が「Success」の場合は、この新しい接続を使用してデータベースに接続します。

接続ナビゲータでの表の参照

7. 接続ナビゲータの「Tables」ノードで、使用可能なオブジェクトを表示します。次の表が存在することを確認します。

COUNTRIES
DEPARTMENTS
EMPLOYEES
JOB_GRADES
JOB_HISTORY
JOBS
LOCATIONS
REGIONS

8. EMPLOYEES表の構造を参照します。
9. DEPARTMENTS表のデータを表示します。

SQL Worksheetの表示

10. 新しいSQL Worksheetをオープンします。SQL Worksheet用に使用できるショートカット・アイコンを確認します。

Oracle Internal & Oracle Academy
Use Only

1

SQL SELECT文を使用したデータの検索

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

目的

この章を終えると、次のことができるようになります。

- SQL SELECT文の機能の列挙
- 基本的なSELECT文の実行

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

データベースからデータを抽出するには、SQL SELECT文を使用する必要があります。ただし、表示する列を制限する必要がある場合があります。この章では、これらのアクションを実行するのに必要なすべてのSQL文について説明します。さらに、繰り返し使用できるSELECT文を作成することもできます。

章の講義項目

- 基本的なSELECT文
- SELECT文の算術式とNULL値
- 列別名
- 連結演算子、リテラル文字列、代替引用符演算子、DISTINCT
キーワードの使用方法
- DESCRIBEコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL SELECT文の機能

投影

表1

選択

表1

結合

表1

表2

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL SELECT文の機能

SELECT文は、データベースから情報を取得します。SELECT文では、次の機能を使用できます。

- **投影**: 問合せによって返される表の列を選択します。必要な数の列だけを選択します。
- **選択**: 問合せによって返される表の行を選択します。様々な基準で、取得する行を制限できます。
- **結合**: 異なる表に格納されているデータ間のリンクを指定して、データを結合します。SQL結合については、「複数の表のデータの表示」の章で説明します。

基本的なSELECT文

```
SELECT *|{[DISTINCT] column|expression [alias],...}  
FROM    table;
```

- SELECTは表示する列を識別します。
- FROMはそれらの列が含まれている表を識別します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

基本的なSELECT文

最も単純な形式のSELECT文には、次の句が含まれている必要があります。

- 表示する列を指定するSELECT句
- SELECT句にリストされた列が含まれている表を識別するFROM句

構文の内容:

SELECT	1つ以上の列のリストです。
*	すべての列を選択します。
DISTINCT	重複を抑制します。
column/expression	名前付きの列または式を選択します。
alias	選択された列に異なるヘッダーを指定します。
FROM table	列が含まれている表を指定します。

注意: このコースを通して、キーワード、句および文という語は次のように使用されます。

- キーワードは個々のSQL要素を表します。
たとえば、SELECTおよびFROMはキーワードです。
- 句はSQL文の一部です。
たとえば、SELECT employee_id, last_nameなどは句です。
- 文は2つ以上の句の組合せです。
たとえば、SELECT * FROM employeesはSQL文です。

すべての列の選択

```
SELECT *  
FROM departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

ORACLE

Copyright © 2007, Oracle. All rights reserved.

すべての列の選択

SELECTキーワードの後にアスタリスク(*)を付けて、表内のデータのすべての列を表示することができます。スライドの例では、部門表に4つの列(DEPARTMENT_ID、DEPARTMENT_NAME、MANAGER_IDおよびLOCATION_ID)が含まれています。また、それぞれの部門に対応する8つの行が含まれています。

SELECTキーワードの後にすべての列を指定して、表内のすべての列を表示することもできます。たとえば、次のSQL文(スライド内の例を参照)は、DEPARTMENTS表の列と行をすべて表示します。

```
SELECT department_id, department_name, manager_id, location_id  
FROM departments;
```

注意: SQL Developerでは、SQL WorksheetにSQL文を入力して、「Execute Statement」アイコンをクリックするか[F9]を押すと文を実行できます。「Results」タブ・ページの出力は、スライドに示すように表示されます。

特定の列の選択

```
SELECT department_id, location_id
FROM departments;
```

	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

ORACLE

Copyright © 2007, Oracle. All rights reserved.

特定の列の選択

SELECT文では、カンマで区切った列名を指定して、表の特定の列を表示することができます。スライドの例では、DEPARTMENTS表のすべての部門番号と所在地番号が表示されています。SELECT句では、出力に表示する順序で列を指定します。たとえば、部門番号の前に所在地を表示する(左から右)には、次の文を使用します。

```
SELECT location_id, department_id
FROM departments;
```

	LOCATION_ID	DEPARTMENT_ID
1	1700	10
2	1800	20
3	1500	50
4	1400	60

...

SQL文の記述

- SQL文は大文字と小文字を区別しません。
- SQL文は1つ以上の行で入力できます。
- キーワードは省略したり、行をまたいで入力したりすることはできません。
- 通常、句は個別の行に入力します。
- インデントを使用して読みやすくします。
- SQL Developerでは、必要に応じてSQL文をセミコロン(;)で終了できます。複数のSQL文を実行する場合は、セミコロンを使用する必要があります。
- SQL*Plusでは、各SQL文の末尾にセミコロン(;)を付ける必要があります。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL文の記述

次の簡単な規則とガイドラインによって、読みやすく編集しやすい有効な文を構築できます。

- SQL文は大文字と小文字を区別しません(指定されていない場合)。
- SQL文は1つ以上の行で入力できます。
- キーワードは、行をまたいで入力したり、省略したりすることはできません。
- 通常、句は、読みやすく編集しやすいように個別の行に入力します。
- コードが読みやすくなるように、インデントを使用する必要があります。
- 通常、キーワードは大文字で入力します。また、表名や列名などの他のすべての語は小文字で入力します。

SQL文の実行

SQL Developerでは、「Run Script」アイコンをクリックするか[F5]を押して、SQL Worksheetでコマンドを実行します。また、「Execute Statement」アイコンをクリックするか[F9]を押して、SQL WorksheetでSQL文を実行することもできます。「Execute Statement」アイコンを使用すると「Enter SQL Statement」ボックス内のマウス・ポインタで示されている文が実行され、「Run Script」アイコンを使用すると「Enter SQL Statement」ボックス内のすべての文が実行されます。「Execute Statement」アイコンでは問合せの出力が「Results」タブ・ページに表示されますが、「Run Script」アイコンではSQL*Plusの表示がエミュレートされて「Script Output」タブ・ページにその出力が表示されます。

SQL*Plusでは、SQL文をセミコロンで終了してから、[Enter]キーを押してコマンドを実行します。

列ヘッダーのデフォルト設定

- SQL Developer:
 - デフォルトのヘッダーの位置合せ: 左詰め
 - デフォルトのヘッダーの表示: 大文字
- SQL*Plus:
 - 文字および日付の列ヘッダーは左詰めにする。
 - 数値の列ヘッダーは右詰めにする。
 - デフォルトのヘッダーの表示: 大文字

ORACLE

Copyright © 2007, Oracle. All rights reserved.

列ヘッダーのデフォルト設定

SQL Developerでは、列ヘッダーは大文字で表示され、左詰めになります。

```
SELECT last_name, hire_date, salary
FROM employees;
```

	LAST_NAME	HIRE_DATE	SALARY
1	King	17-JUN-87	24000
2	Kochhar	21-SEP-89	17000
3	De Haan	13-JAN-93	17000
4	Hunold	03-JAN-90	9000
5	Ernst	21-MAY-91	6000
6	Lorentz	07-FEB-99	4200
7	Mourgos	16-NOV-99	5800
8	Rajs	17-OCT-95	3500

...

列ヘッダーは別名で表示されるように上書きできます。列の別名については、この章で後述します。

章の講義項目

- 基本的なSELECT文
- SELECT文の算術式とNULL値
- 列別名
- 連結演算子、リテラル文字列、代替引用符演算子、DISTINCT
キーワードの使用方法
- DESCRIBEコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

算術式

算術演算子を使用して、数値と日付データを使用した式を作成します。

演算子	説明
+	加算
-	減算
*	乗算
/	除算

ORACLE

Copyright © 2007, Oracle. All rights reserved.

算術式

データの表示方法は変更することができます。また、計算したり、what-ifシナリオを確認したりすることもできます。これらはすべて算術式を使用して実行できます。算術式には、列名、定数値および算術演算子を使用できます。

算術演算子

スライドに、SQLで使用できる算術演算子がリストされています。算術演算子は、SQL文の任意の句 (FROM句を除く) で使用できます。

注意: DATEおよびTIMESTAMPデータ型では、加算演算子と減算演算子ののみを使用できます。

算術演算子の使用方法

```
SELECT last_name, salary, salary + 300
FROM employees;
```

	LAST_NAME	SALARY	SALARY+300
1	King	24000	24300
2	Kochhar	17000	17300
3	De Haan	17000	17300
4	Hunold	9000	9300
5	Ernst	6000	6300
6	Lorentz	4200	4500
7	Mourgos	5800	6100
8	Rajs	3500	3800
9	Davies	3100	3400
10	Matos	2600	2900

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

算術演算子の使用方法

スライドの例では、加算演算子を使用して、すべての従業員に対して\$300を増額した場合の給与を計算しています。スライドの出力には、SALARY+300の列が表示されています。

計算結果の列SALARY+300はEMPLOYEES表の新しい列ではないことに注意してください。これは表示目的にのみ使用されます。デフォルトでは、新しい列の名前はその列を生成した計算に基づきます。この場合はsalary+300です。

注意: Oracleサーバーは、算術演算子の前後にある空白を無視します。

演算子の優先順位

算術式に複数の演算子が含まれている場合は、乗算と除算が最初に計算されます。式内の演算子が同じ優先順位の場合は、左から右へ計算されます。

カッコを使用して、カッコ内の式が最初に計算されるように強制できます。

優先順位の規則:

- 乗算および除算は、加算および減算よりも先に実行されます。
- 優先順位が同じ演算子は、左から右へと計算されます。
- カッコを使用すると、デフォルトの優先順位を無視したり、文を明確にしたりできます。

演算子の優先順位

```
SELECT last_name, salary, 12*salary+100
FROM employees;
```

1

	LAST_NAME	SALARY	12*SALARY+100
1	King	24000	288100
2	Kochhar	17000	204100
3	De Haan	17000	204100

...

```
SELECT last_name, salary, 12*(salary+100)
FROM employees;
```

2

	LAST_NAME	SALARY	12*(SALARY+100)
1	King	24000	289200
2	Kochhar	17000	205200
3	De Haan	17000	205200

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演算子の優先順位(続き)

スライドの最初の例では、従業員の姓、給与および年間所得が表示されています。月給に12を乗算して、1回の賞与金額\$100を加えることで、年間所得が計算されます。乗算は加算の前に実行されることに注意してください。

注意: カッコを使用すると、標準の優先順位を強調し、記述を明確にすることができます。たとえば、スライド内の式は $(12*salary)+100$ と記述することができます。この場合、結果は変更されません。

カッコの使用法

カッコを使用すると、優先順位の規則を無視して、演算子の実行順序を任意に指定することができます。

スライドの2番目の例では、従業員の姓、給与および年間所得が表示されています。年間所得は、月給に毎月の賞与金額\$100を加え、その小計に12を乗算して計算されます。カッコが使用されているため、加算の方が乗算よりも優先されます。

NULL値の定義

- NULLは、使用できない値、割り当てられていない値、不明な値、または適用できない値を表します。
- NULLは、ゼロまたは空白とは異なります。

```
SELECT last_name, job_id, salary, commission_pct
FROM employees;
```

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	King	AD_PRES	24000	(null)
2	Kochhar	AD_VP	17000	(null)
...				
12	Zlotkey	SA_MAN	10500	0.2
13	Abel	SA_REP	11000	0.3
14	Taylor	SA_REP	8600	0.2
...				
19	Higgins	AC_MGR	12000	(null)
20	Gietz	AC_ACCOUNT	8300	(null)

ORACLE

Copyright © 2007, Oracle. All rights reserved.

NULL値の定義

行の特定の列に対するデータ値が欠落している場合、その値はNULLである、またはNULLを含むといいます。

NULLは、使用できない値、割り当てられていない値、不明な値、または適用できない値を表します。NULLはゼロまたは空白とは異なります。ゼロは数値で、空白は文字です。

任意のデータ型の列にNULLを含めることができます。ただし、一部の制約 (NOT NULLおよびPRIMARY KEY) によって、NULLを列に使用できない場合があります。

EMPLOYEES表のCOMMISSION_PCT列で、販売マネージャまたは販売担当者のみが歩合を受け取ることができることに注意してください。他の従業員には歩合を受け取る権限がありません。NULLは、このことを示しています。

注意: デフォルトでは、SQL Developerはリテラル (null) を使用してNULL値を識別します。ただし、この設定を、環境により適した設定に変更することができます。これを行うには、「Tools」メニューから「Preferences」を選択します。「Preferences」ダイアログ・ボックスで、「Database」ノードを展開します。「Advanced Parameters」をクリックして、右ペインの「Display Null value As」に適切な値を入力します。

算術式のNULL値

NULL値が含まれている算術式の結果はNULLになります。

```
SELECT last_name, 12*salary*commission_pct
FROM employees;
```

	LAST_NAME	12*SALARY*COMMISSION_PCT
1	King	(null)
2	Kochhar	(null)

...

12	Zlotkey	25200
13	Abel	39600
14	Taylor	20640

...

19	Higgins	(null)
20	Gietz	(null)

ORACLE

Copyright © 2007, Oracle. All rights reserved.

算術式のNULL値

算術式のいずれかの列の値がNULLの場合、結果はNULLになります。たとえば、ゼロで除算を実行した場合は、エラーになります。ただし、NULLで数値を除算した場合は、結果はNULLまたは不明になります。

スライドの例では、従業員Kingは歩合を受け取っていません。算術式のCOMMISSION_PCT列がNULLであるため、結果もNULLになります。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のOracle SQLの基本要素に関する節を参照してください。

章の講義項目

- 基本的なSELECT文
- SELECT文の算術式とNULL値
- 列別名
- 連結演算子、リテラル文字列、代替引用符演算子、DISTINCT
キーワードの使用方法
- DESCRIBEコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

列別名の定義

列別名の概要は次のとおりです。

- 列ヘッダーの名前を変更します。
- 計算に便利です。
- 列名の直後に指定します(必要に応じて、列名と列別名の間にASキーワードを使用することもできます)。
- 空白や特殊文字を含める場合、または大文字と小文字を区別する場合は、二重引用符が必要です。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

列別名の定義

通常、SQL Developerでは、問合せの結果を表示するときに選択した列の名前が列ヘッダーとして使用されます。このヘッダーは、説明的ではないため、わかりにくい場合があります。列別名を使用して列ヘッダーを変更することができます。

空白をセパレータとして使用して、SELECTリストの列の後に別名を指定します。デフォルトでは、別名のヘッダーは大文字で表示されます。別名にスペースや特殊文字(#や\$など)を含める場合、または大文字と小文字を区別する場合は、別名を二重引用符(“ ”)で囲みます。

列別名の使用方法

```
SELECT last_name AS name, commission_pct comm
FROM employees;
```

	NAME	COMM
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)

...

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;
```

	Name	Annual Salary
1	King	288000
2	Kochhar	204000
3	De Haan	204000

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

列別名の使用方法

最初の例では、すべての従業員の名前と歩合の割合が表示されています。オプションのASキーワードが列別名の前に指定されていることに注意してください。問合せの結果は、ASキーワードが使用されているかどうかに関係なく同じ結果になります。また、SQL文では列別名 (nameとcomm) に小文字が使用されていますが、問合せの結果には列ヘッダーが大文字で表示されていることにも注意してください。前のスライドで説明したように、デフォルトで列ヘッダーは大文字で表示されます。

2番目の例では、すべての従業員の姓と年間所得が表示されています。Annual Salaryには空白が含まれているため、二重引用符で囲まれています。出力の列ヘッダーが列別名とまったく同じであることに注意してください。

章の講義項目

- 基本的なSELECT文
- SELECT文の算術式とNULL値
- 列別名
- 連結演算子、リテラル文字列、代替引用符演算子、DISTINCT
キーワードの使用方法
- DESCRIBEコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

連結演算子

連結演算子には次の特徴があります。

- 列または文字列を他の列にリンクします。
- 2本の縦棒(||)で表されます。
- 文字式の結果の列が作成されます。

```
SELECT last_name || job_id AS "Employees"
FROM employees;
```

	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG
5	FayMK_REP

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

連結演算子

連結演算子(||)を使用すると、列を他の列、算術式または定数値にリンクして、文字式を作成できます。演算子の両側の列が結合されて、1つの列が出力されます。

この例では、LAST_NAMEとJOB_IDが連結されて、別名のEmployeesが表示されています。従業員の姓と職務コードが結合されて1つの列に出力されていることに注意してください。

別名の前にASキーワードを指定すると、SELECT句が読みやすくなります。

NULL値と連結演算子

NULL値と文字列を連結した場合、結果は文字列になります。したがって、LAST_NAME || NULLの結果は、LAST_NAMEになります。

注意: 日付の式を他の式または列に連結することもできます。

リテラル文字列

- リテラルは、SELECT文に含まれる文字、数値または日付です。
- 日付および文字リテラル値は、一重引用符で囲む必要があります。
- 各文字列は、行が戻されるたびに1回出力されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

リテラル文字列

リテラルは、SELECTリストに含まれる文字、数値または日付です。列名または列別名ではありません。行が戻されるたびに出力されます。自由書式テキストのリテラル文字列は、問合せ結果に含めることができ、SELECTリスト内の列と同様に扱われます。

日付および文字リテラルは、一重引用符(' ')で囲む必要があります。数値リテラルは一重引用符で囲む必要はありません。

リテラル文字列の使用方法

```
SELECT last_name || ' is a ' || job_id
       AS "Employee Details"
FROM   employees;
```

Employee Details	
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
...	
18	Vargas is a ST_CLERK
19	Whalen is a AD_ASST
20	Zlotkey is a SA_MAN

ORACLE

Copyright © 2007, Oracle. All rights reserved.

リテラル文字列の使用方法

スライドの例では、すべての従業員の姓と職務コードが表示されています。列のヘッダーは Employee Details です。SELECT 文の一重引用符の間にある空白に注意してください。これらの空白によって、出力が読みやすくなっています。

次の例では、各従業員の姓と給与がリテラルで連結されて、さらにわかりやすい行が戻されます。

```
SELECT last_name || ': 1 Month salary = ' || salary Monthly
FROM   employees;
```

MONTHLY	
1	King: 1 Month salary = 24000
2	Kochhar: 1 Month salary = 17000
3	De Haan: 1 Month salary = 17000
4	Hunold: 1 Month salary = 9000
5	Ernst: 1 Month salary = 6000
6	Lorentz: 1 Month salary = 4200
7	Mourgos: 1 Month salary = 5800
8	Rajs: 1 Month salary = 3500

...

代替引用符(q)演算子

- 独自の引用符デリミタを指定します。
- 任意のデリミタを選択します。
- さらに読みやすく、使いやすくなります。

```
SELECT department_name || ' Department' ||  
       q'['s Manager Id: ]'  
       || manager_id  
       AS "Department and Manager"  
FROM departments;
```

	Department and Manager
1	Administration Department's Manager Id:200
2	Marketing Department's Manager Id:201
3	Shipping Department's Manager Id:124
4	IT Department's Manager Id:103
5	Sales Department's Manager Id:149
6	Executive Department's Manager Id:100
7	Accounting Department's Manager Id:205
8	Contracting Department's Manager Id:

ORACLE

Copyright © 2007, Oracle. All rights reserved.

代替引用符(q)演算子

多くのSQL文では、式または条件に文字リテラルが使用されます。リテラル自体に一重引用符が含まれる場合は、引用符(q)演算子を使用して、独自の引用符デリミタを選択できます。

任意の有効なデリミタとして、シングルスバイト文字、マルチバイト文字、または対の文字([], { }, () または < >)を選択できます。

スライドの例では、通常は文字列のデリミタとして解析される一重引用符が文字列に含まれています。ただし、q演算子を使用することで、角カッコ[]が引用符デリミタとして使用されています。角カッコのデリミタに囲まれた文字列は、リテラル文字列として解析されます。

重複行

問合せでは、デフォルトで、重複行を含めたすべての行が表示されます。

```
SELECT department_id  
FROM employees;
```

①

	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60

...

```
SELECT DISTINCT department_id  
FROM employees;
```

②

	DEPARTMENT_ID
1	(null)
2	90
3	20
4	110

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

重複行

特に指定しない場合、SQLでは重複行を除外せずに問合せの結果が表示されます。スライドの最初の例では、EMPLOYEES表のすべての部門番号が表示されます。部門番号が繰り返し表示されていることに注意してください。

重複行を結果から除外するには、SELECT句のSELECTキーワードの直後にDISTINCTキーワードを指定します。スライドの2番目の例では、EMPLOYEES表には実際には20行の行が含まれていますが、一意の部門番号が7つだけ表示されています。

DISTINCT修飾子の後には複数の列を指定できます。DISTINCT修飾子は、選択されているすべての列に適用され、重複していない列の組合せがすべて結果に表示されます。

```
SELECT DISTINCT department_id, job_id  
FROM employees;
```

	DEPARTMENT_ID	JOB_ID
1	110	AC_ACCOUNT
2	90	AD_VP
3	50	ST_CLERK
4	80	SA_REP
5	50	ST_MAN

...

章の講義項目

- 基本的なSELECT文
- SELECT文の算術式とNULL値
- 列別名
- 連結演算子、リテラル文字列、代替引用符演算子、DISTINCT
キーワードの使用方法
- DESCRIBEコマンド

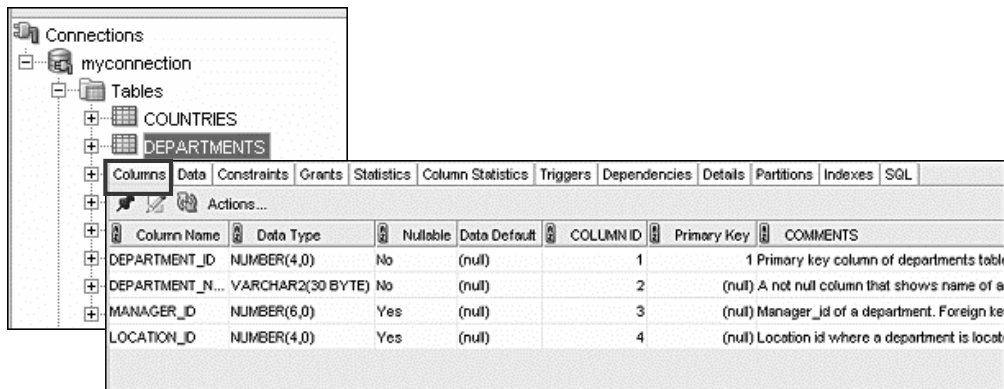
ORACLE

Copyright © 2007, Oracle. All rights reserved.

表構造の表示

- DESCRIBEコマンドを使用して、表の構造を表示します。
- または、「Connections」ツリーで表を選択し、「Columns」タブを使用して表の構造を表示します。

```
DESC[RIBE] tablename
```



Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
DEPARTMENT_ID	NUMBER(4,0)	No	(null)	1		1 Primary key column of departments table
DEPARTMENT_N...	VARCHAR2(30 BYTE)	No	(null)	2		(null) A not null column that shows name of a
MANAGER_ID	NUMBER(6,0)	Yes	(null)	3		(null) Manager_id of a department. Foreign ke
LOCATION_ID	NUMBER(4,0)	Yes	(null)	4		(null) Location id where a department is locat

ORACLE

Copyright © 2007, Oracle. All rights reserved.

表構造の表示

SQL Developerでは、DESCRIBEコマンドを使用して表の構造を表示できます。このコマンドは列名とデータ型を表示し、列にデータが含まれている必要があるかどうか(つまり、列にNOT NULL制約があるかどうか)を示します。

構文に含まれる *table name* は、ユーザーがアクセスできる既存の表、ビューまたはシノニムの名前です。

SQL Developer GUIインタフェースを使用して、「Connections」ツリーで表を選択し、「Columns」タブを使用して表の構造を表示できます。

注意: DESCRIBEコマンドは、SQL*PlusとSQL Developerの両方でサポートされています。

DESCRIBEコマンドの使用方法

```
DESCRIBE employees
```

```
DESCRIBE employees
Name                               Null    Type
-----
EMPLOYEE_ID                       NOT NULL NUMBER(6)
FIRST_NAME                        VARCHAR2(20)
LAST_NAME                         NOT NULL VARCHAR2(25)
EMAIL                             NOT NULL VARCHAR2(25)
PHONE_NUMBER                      VARCHAR2(20)
HIRE_DATE                         NOT NULL DATE
JOB_ID                            NOT NULL VARCHAR2(10)
SALARY                            NUMBER(8,2)
COMMISSION_PCT                    NUMBER(2,2)
MANAGER_ID                        NUMBER(6)
DEPARTMENT_ID                     NUMBER(4)

11 rows selected
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

DESCRIBEコマンドの使用方法

結果の表示では、*Null*は、この列の値が不明な可能性があることを示しています。NOT NULLは、列にデータが含まれている必要があることを示しています。*Type*は、列のデータ型を表示します。データ型について、次の表に説明します。

データ型	説明
NUMBER(<i>p</i> , <i>s</i>)	最大桁数 <i>p</i> 、小数点以下の桁数 <i>s</i> の数値
VARCHAR2(<i>s</i>)	最大サイズ <i>s</i> の可変長文字値
DATE	January 1, 4712 B.C.～December 31, A.D. 9999の範囲の日時の値
CHAR(<i>s</i>)	サイズ <i>s</i> の固定長文字値

まとめ

この章では、次のことを学習しました。

- 次の処理を行うSELECT文の記述
 - 表からすべての行と列を戻す
 - 表から指定された行を戻す
 - 列別名を使用して、さらにわかりやすい列ヘッダーを表示する

```
SELECT *|{ [DISTINCT] column|expression [alias],...}  
FROM table;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SELECT文

この章では、SELECT文を使用してデータベース表からデータを取得する方法を学習しました。

```
SELECT *|{ [DISTINCT] column [alias],...}  
FROM table;
```

構文の内容:

SELECT	1つ以上の列のリストです。
*	すべての列を選択します。
DISTINCT	重複を抑制します。
column/expression	名前付きの列または式を選択します。
alias	選択された列に異なるヘッダーを指定します。
FROM table	列が含まれている表を指定します。

演習1: 概要

この演習では次の項目について説明しています。

- 異なる表のすべてのデータの選択
- 表の構造の説明
- 算術演算の実行と列名の指定

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習1: 概要

この演習では、単純なSELECT問合せを記述します。問合せには、この章で学習したほとんどのSELECT句と演算が含まれます。

演習1

パート1

知識について確認します。

1. 次のSELECT文は正しく実行されます。

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

○/×

2. 次のSELECT文は正しく実行されます。

```
SELECT *
FROM   job_grades;
```

○/×

3. 次の文には4つのコーディング・エラーがあります。特定できますか？

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

パート2

演習を始める前に次の点を確認します。

- すべての演習ファイルをD:\labs\SQL1\labsに保存します。
- SQL文をSQL Worksheetに入力します。SQL Developerでスクリプトを保存するには、必要なSQL Worksheetがアクティブになっていることを確認し、「File」メニューから「Save As」を選択するか、またはSQL Worksheet内で右クリックし、「Save file」を選択してSQL文をlab_<lessonno>_<stepno>.sqlスクリプトとして保存します。既存のスクリプトを変更するときは、必ず「Save As」を使用して、異なるファイル名で保存してください。
- 問合せを実行するには、SQL Worksheetで「Execute Statement」アイコンをクリックします。または、[F9]を押すこともできます。DML文およびDDL文の場合は、「Run Script」アイコンを使用するか、[F5]を押します。
- 問合せを実行した後は、同じワークシートに次の問合せを入力しないようにします。新しいワークシートを開いてください。

あなたはAcme CorporationのSQLプログラマとして採用されました。最初の仕事は、人事管理表のデータに基づいたレポートの作成です。

4. まず、DEPARTMENTS表の構造とその内容を確認します。

DESCRIBE departments		
Name	Null	Type

DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)
4 rows selected		

演習1(続き)

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

5. EMPLOYEES表の構造を確認する必要があります。

DESCRIBE employees		
Name	Null	Type
-----	-----	-----
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)
11 rows selected		

HR部門は、各従業員の姓、職務コード、雇用日、従業員番号を表示する問合せを必要としています。また、従業員番号を最初に表示する必要があります。HIRE_DATE列に別名 STARTDATEを指定します。SQL文を、lab_01_05.sqlという名前のファイルに保存して、このファイルをHR部門にディスパッチできるようにします。

演習1(続き)

6. lab_01_05.sqlファイルで問合せをテストして、正しく実行されることを確認します。

注意: 問合せを実行した後は、同じワークシートに次の問合せを入力しないようにします。
新しいワークシートを開いてください。

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	100	King	AD_PRES	17-JUN-87
2	101	Kochhar	AD_VP	21-SEP-89
3	102	De Haan	AD_VP	13-JAN-93
4	103	Hunold	IT_PROG	03-JAN-90
5	104	Ernst	IT_PROG	21-MAY-91
6	107	Lorentz	IT_PROG	07-FEB-99
7	124	Mourgos	ST_MAN	16-NOV-99
8	141	Rajs	ST_CLERK	17-OCT-95
9	142	Davies	ST_CLERK	29-JAN-97
10	143	Matos	ST_CLERK	15-MAR-98

...

19	205	Higgins	AC_MGR	07-JUN-94
20	206	Gietz	AC_ACCOUNT	07-JUN-94

7. HR部門は、EMPLOYEES表から一意の職務コードをすべて表示する問合せを必要としています。

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	IT_PROG
7	MK_MAN
8	MK_REP
9	SA_MAN
10	SA_REP
11	ST_CLERK
12	ST_MAN

演習1(続き)

パート3

時間があるときは、次の演習問題に進みます。

- HR部門は、従業員に関するそのレポートに、さらにわかりやすい列ヘッダーを使用するように求めています。文をlab_01_05.sqlから新しいSQL Worksheetにコピーします。列ヘッダーにそれぞれEmp #、Employee、Job、Hire Dateという名前を付けます。その後、問合せを再度実行します。

	A2	Emp #	A2	Employee	A2	Job	Hire Date
1		100	King		AD_PRES	17-JUN-87	
2		101	Kochhar		AD_VP	21-SEP-89	
3		102	De Haan		AD_VP	13-JAN-93	
4		103	Hunold		IT_PROG	03-JAN-90	
5		104	Ernst		IT_PROG	21-MAY-91	
6		107	Lorentz		IT_PROG	07-FEB-99	
7		124	Mourgos		ST_MAN	16-NOV-99	
8		141	Rajs		ST_CLERK	17-OCT-95	
9		142	Davies		ST_CLERK	29-JAN-97	
10		143	Matos		ST_CLERK	15-MAR-98	

...

19		205	Higgins		AC_MGR	07-JUN-94
20		206	Gietz		AC_ACCOUNT	07-JUN-94

- HR部門は、すべての従業員とその職務IDのレポートを要求しています。姓に職務IDを(カンマおよび空白で区切って)連結して表示し、列にEmployee and Titleという名前を付けます。

	A2	Employee and Title
1		Abel, SA_REP
2		Davies, ST_CLERK
3		De Haan, AD_VP
4		Ernst, IT_PROG
5		Fay, MK_REP
6		Gietz, AC_ACCOUNT
7		Grant, SA_REP
8		Hartstein, MK_MAN
9		Higgins, AC_MGR
10		Hunold, IT_PROG

...

19		Whalen, AD_ASST
20		Zlotkey, SA_MAN

演習1(続き)

さらに演習を続ける場合は、次の演習問題に進みます。

10. EMPLOYEES表のデータを理解するために、その表のすべてのデータを表示する問合せを作成します。それぞれの列の出力をカンマで区切ります。列タイトルにTHE_OUTPUTという名前を付けます。

Results	Script Output	Explain	Autotrace	DBMS Output	OWA Output
Results:					
THE_OUTPUT					
1	100	Steven	King	SKING	515.123.4567,AD_PRES,,17-JUN-87,24000,,90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568,AD_VP,100,21-SEP-89,17000,,90
3	102	Lex	De Haan	LDEHAAN	515.123.4569,AD_VP,100,13-JAN-93,17000,,90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567,IT_PROG,102,03-JAN-90,9000,,60
5	104	Bruce	Ernst	BERNST	590.423.4568,IT_PROG,103,21-MAY-91,6000,,60
6	107	Diana	Lorentz	DLORENTZ	590.423.5567,IT_PROG,103,07-FEB-99,4200,,60
7	124	Kevin	Mourgos	KMOURGOS	650.123.5234,ST_MAN,100,16-NOV-99,5800,,50
8	141	Trenna	Rajs	TRAJS	650.121.8009,ST_CLERK,124,17-OCT-95,3500,,50
9	142	Curtis	Davies	CDAVIES	650.121.2994,ST_CLERK,124,29-JAN-97,3100,,50
10	143	Randall	Matos	RMATOS	650.121.2874,ST_CLERK,124,15-MAR-98,2600,,50

...

19	205	Shelley	Higgins	SHIGGINS	515.123.8080,AC_MGR,101,07-JUN-94,12000,,110
20	206	William	Gietz	WGIETZ	515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110

データの制限とソート

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

目的

この章を終えると、次のことができるようになります。

- 問合せで取得する行の制限
- 問合せで取得する行のソート
- アンパサンド置換を使用した実行時の出力の制限とソート

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

データベースからデータを取得するとき、次の操作が必要になる場合があります。

- 表示されるデータの行数の制限
- 行の表示順序の指定

この章では、これらのアクションの実行に使用するSQL文について説明します。

章の講義項目

- 行の制限に使用する機能
 - WHERE句
 - =、<=、BETWEEN、IN、LIKEおよびNULL条件を使用した比較条件
 - AND、ORおよびNOT演算子を使用した論理条件
- 式内の演算子の優先順位の規則
- ORDER BY句を使用した行のソート
- 置換変数
- DEFINEおよびVERIFYコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

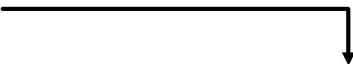
選択を使用した行の制限

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90
4	103	Hunold	IT_PROG	60
5	104	Ernst	IT_PROG	60
6	107	Lorentz	IT_PROG	60

...

“部門90の
すべての従業員を
取得”



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

ORACLE

Copyright © 2007, Oracle. All rights reserved.

選択を使用した行の制限

スライドの例では、部門90に所属するすべての従業員を表示することを想定しています。DEPARTMENT_ID列の値が90である行のみが戻されます。このような制限方法が、SQLのWHERE句の基礎です。

選択される行の制限

- 次のように、WHERE句を使用して戻される行を制限します。

```
SELECT *|{[DISTINCT] column|expression [alias],...}  
FROM table  
[WHERE condition(s)];
```

- FROM句の後にWHERE句を指定します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

選択される行の制限

WHERE句を使用して、問合せから戻される行を制限できます。WHERE句には一致する必要がある条件を入力し、WHERE句はFROM句の直後に指定します。条件がTRUEの場合、その条件に一致する行が戻されます。

構文の内容

WHERE 問合せを、条件に一致する行に制限します。

condition 列名、式、定数および比較演算子で構成されます。条件には1つ以上の式と論理(ブール)演算子の組合せを指定します。TRUE、FALSEまたはUNKNOWNの値が戻されます。

WHERE句では、列、リテラル、算術式または関数の値を比較できます。次の3つの要素で構成されます。

- 列名
- 比較条件
- 列名、定数または値のリスト

WHERE句の使用方法

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

ORACLE

Copyright © 2007, Oracle. All rights reserved.

WHERE句の使用方法

この例では、部門90に所属するすべての従業員の従業員ID、姓、職務ID、部門番号をSELECT文で取得します。

注意: WHERE句では、列の別名を使用できません。

文字列と日付

- 文字列および日付値は、一重引用符で囲みます。
- 文字値は大文字と小文字が区別され、日付値は書式が区別されます。
- デフォルトの日付の表示書式は、DD-MON-RRです。

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-FEB-96' ;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

文字列と日付

WHERE句内の文字列および日付は、一重引用符('')で囲む必要があります。ただし、数値定数は一重引用符で囲む必要はありません。

すべての文字検索で大文字と小文字が区別されます。次の例では、行が戻されません。これは、EMPLOYEES表ではすべての姓が大文字と小文字が混在した状態で格納されているためです。

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'WHALEN';
```

Oracle Databaseでは、世紀、年、月、日、時、分および秒を表す内部的な数値書式で日付が格納されます。デフォルトの日付の表示書式は、DD-MON-RRです。

注意: RR書式の詳細とデフォルトの日付書式の変更については、単一行関数を使用した出力のカスタマイズに関する章を参照してください。また、同じ章で、UPPERやLOWERなどの単一行関数を使用して大文字と小文字の区別を無視する方法についても学習します。

比較演算子

演算子	意味
=	等しい
>	大きい
>=	以上
<	小さい
<=	以下
<>	等しくない
BETWEEN ...AND...	2つの値の間(その値自体を含む)
IN(set)	値のリストのいずれかに一致
LIKE	文字のパターンに一致
IS NULL	NULL値である

ORACLE

Copyright © 2007, Oracle. All rights reserved.

比較演算子

比較演算子は、1つの式を別の値または式と比較する条件で使います。比較演算子はWHERE句で次の書式で使います。

構文

... WHERE *expr operator value*

例

```
... WHERE hire_date = '01-JAN-95'  
... WHERE salary >= 6000  
... WHERE last_name = 'Smith'
```

WHERE句では、別名を使用できません。

注意: 記号!=および<>でも、「等しくない」条件を表すことができます。

比較演算子の使用方法

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

ORACLE

Copyright © 2007, Oracle. All rights reserved.

比較演算子の使用方法

この例では、SELECT文で、EMPLOYEES表から給与が\$3,000以下の従業員の姓と給与を取得します。WHERE句に明示的な値が指定されていることに注意してください。明示的な値3000は、EMPLOYEES表のSALARY列内の給与値と比較されます。

BETWEEN演算子を使用した範囲条件

BETWEEN演算子は、次のように、値の範囲に基づいて行を表示するために使用します。

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500 ;
```

↑
下限

↑
上限

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

ORACLE

Copyright © 2007, Oracle. All rights reserved.

BETWEEN演算子を使用した範囲条件

BETWEEN演算子を使用すると、値の範囲に基づいて行を表示できます。指定する範囲には、上限と下限が含まれます。

スライドの例のSELECT文は、EMPLOYEES表から給与が\$2,500～\$3,500の範囲にある従業員の行を戻します。

BETWEEN演算子で指定した範囲には、指定した値自体も含まれます。ただし、最初に下限を指定する必要があります。

文字値にもBETWEEN演算子を使用できます。

```
SELECT last_name
FROM employees
WHERE last_name BETWEEN 'King' AND 'Smith';
```

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos
6	Rajs

IN演算子を使用したメンバーシップ条件

IN演算子は、次のように、リスト内の値に関するテストを行うために使用します。

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

ORACLE

Copyright © 2007, Oracle. All rights reserved.

IN演算子を使用したメンバーシップ条件

指定した値セットの値に関してテストするには、IN演算子を使用します。IN演算子を使用して定義された条件は、メンバーシップ条件とも呼ばれます。

スライドの例では、従業員のマネージャの従業員番号が100、101または201である場合に、その従業員すべての従業員番号、姓、給与およびマネージャの従業員番号が表示されます。

IN演算子では任意のデータ型を使用できます。次の例では、姓がWHERE句の名前リストに含まれている従業員の行が、EMPLOYEES表から戻されます。

```
SELECT employee_id, manager_id, department_id
FROM   employees
WHERE  last_name IN ('Hartstein', 'Vargas');
```

リスト内に文字または日付を指定する場合は、一重引用符(')で囲む必要があります。

注意: IN演算子は、Oracleサーバーで内部的に、a=value1またはa=value2またはa=value3などのOR条件のセットとして評価されます。したがって、IN演算子を使用してもパフォーマンス上の利点はありません。IN演算子は、論理的単純性のみを目的として使用されます。

LIKE演算子を使用したパターン一致

- LIKE演算子は、有効な検索文字列値のワイルドカード検索を実行するために使用します。
- 検索条件には、次のようなリテラルな文字または数値を含めることができます。
 - %は0個以上の文字を表す。
 - _は1個の文字を表す。

```
SELECT first_name
FROM employees
WHERE first_name LIKE 'S%';
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

LIKE演算子を使用したパターン一致

検索する正確な値がわからない場合もあります。LIKE演算子を使用すると、文字パターンに一致する行を選択できます。文字パターン一致操作は、ワイルドカード検索と呼ばれます。検索文字列の構成には、2つの記号を使用できます。

記号	説明
%	0個以上の一連の文字を表す
_	1個の文字を表す

スライドのSELECT文では、名前が文字“S”で始まる従業員の姓がEMPLOYEES表から戻されます。“S”が大文字であることに注意してください。このように指定されているため、小文字の“s”で始まる名前は戻されません。

LIKE演算子は、一部のBETWEEN比較のショートカットとして使用できます。次の例では、1995年1月～1995年12月に入社したすべての従業員の姓と雇用日が表示されます。

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date LIKE '%95';
```


ワイルドカード文字の組合せ

- パターン一致では、次のように2つのワイルドカード文字(%,_)とリテラル文字を組み合わせることができます。

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%' ;
```

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

- ESCAPE 識別子を使用することで、実際の%および_記号を検索できます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ワイルドカード文字の組合せ

%および_記号は、リテラル文字と任意に組み合わせて使用できます。スライドの例では、姓の2番目の文字として文字“o”が含まれているすべての従業員の姓が表示されます。

ESCAPE識別子

実際の%文字および_文字の完全一致を検索する必要がある場合は、ESCAPE識別子を使用します。このオプションは、エスケープ文字として使用する文字を指定します。SA_が含まれている文字列を検索する場合には、次のSQL文を使用できます。

```
SELECT employee_id, last_name, job_id
FROM   employees WHERE  job_id LIKE '%SA\__%' ESCAPE '\';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID
1	149	Zlotkey	SA_MAN
2	174	Abel	SA_REP
3	176	Taylor	SA_REP
4	178	Grant	SA_REP

ESCAPE識別子は、バックスラッシュ(\)をエスケープ文字として識別します。このSQL文では、エスケープ文字がアンダースコア(_)の前に指定されています。これによって、Oracleサーバーはアンダースコアをリテラルとして解釈します。

NULL条件の使用方法

IS NULL演算子は、NULLに関するテストを行うために使用します。

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

	LAST_NAME	MANAGER_ID
1	King	(null)

ORACLE

Copyright © 2007, Oracle. All rights reserved.

NULL条件の使用方法

NULL条件には、IS NULL条件とIS NOT NULL条件があります。

IS NULL条件では、NULLに関するテストが行われます。NULL値は、値が使用できない、割り当てられていない、不明である、または適用不可能であることを表します。したがって、=ではテストできません。NULLはどのような値に対しても等しいまたは等しくないとは評価できないためです。スライドの例では、担当マネージャがいないすべての従業員の姓とマネージャを取得しています。例をもう1つ示します。歩合給を受け取る権利がないすべての従業員の姓、職務IDおよび歩合を表示するには、次のSQL文を使用します。

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

	LAST_NAME	JOB_ID	COMMISSION_PCT
1	King	AD_PRES	(null)
2	Kochhar	AD_VP	(null)

...

15	Higgins	AC_MGR	(null)
16	Gietz	AC_ACCOUNT	(null)

論理演算子を使用した条件の定義

演算子	意味
AND	コンポーネントの条件が両方とも真の場合にTRUEを返す
OR	コンポーネントの条件のどちらかが真の場合にTRUEを返す
NOT	条件が偽の場合にTRUEを返す

ORACLE

Copyright © 2007, Oracle. All rights reserved.

論理演算子を使用した条件の定義

論理条件は、2つの構成要素の条件の結果を組み合わせ、それらの条件に基づいて単一の結果を生成するか、または単一の条件の結果を反転させます。条件の総合結果がTRUEの場合にのみ行が戻されます。

SQLでは、次の3つの論理演算子を使用できます。

- AND
- OR
- NOT

これまでのすべての例では、WHERE句に1つの条件のみを指定していました。ANDおよびOR演算子を使用すると、1つのWHERE句に複数の条件を指定することができます。

AND演算子の使用方法

ANDでは、両方の要素条件がTRUEになる必要があります。

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	149	Zlotkey	SA_MAN	10500
2	201	Hartstein	MK_MAN	13000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

AND演算子の使用方法

この例では、選択する任意のレコードで両方の構成要素条件がTRUEになる必要があります。したがって、職務に文字列‘MAN’が含まれ、給与が\$10,000以上の従業員のみが選択されます。すべての文字検索で大文字と小文字が区別されます。つまり、‘MAN’が大文字でない場合は、行が戻されません。また、文字列は引用符で囲む必要があります。

AND真理値表

次の表は、2つの式をANDで結合した結果を示しています。

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR演算子の使用方法

ORでは、どちらかの要素条件がTRUEになる必要があります。

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	100 King	AD_PRES	24000
2	101 Kochhar	AD_VP	17000
3	102 De Haan	AD_VP	17000
4	124 Mourgos	ST_MAN	5800
5	149 Zlotkey	SA_MAN	10500
6	174 Abel	SA_REP	11000
7	201 Hartstein	MK_MAN	13000
8	205 Higgins	AC_MGR	12000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

OR演算子の使用方法

この例では、選択する任意のレコードでどちらかの構成要素条件がTRUEになることができます。したがって、職務IDに文字列'MAN'が含まれるか、または給与が\$10,000以上の従業員が選択されます。

OR真理値表

次の表は、2つの式をORで結合した結果を示しています。

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT演算子の使用方法

```
SELECT last_name, job_id
FROM employees
WHERE job_id
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

ORACLE

Copyright © 2007, Oracle. All rights reserved.

NOT演算子の使用方法

スライドの例では、職務IDがIT_PROG、ST_CLERKまたはSA_REPではないすべての従業員の姓と職務IDが表示されます。

NOT真理値表

次の表は、条件にNOT演算子を適用した結果を示しています。

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

注意: NOT演算子は、BETWEEN、LIKE、NULLなどの他のSQL演算子と一緒に使用することもできます。

```
... WHERE job_id      NOT IN ('AC_ACCOUNT', 'AD_VP')
... WHERE salary      NOT BETWEEN 10000 AND 15000
... WHERE last_name   NOT LIKE '%A%'
... WHERE commission_pct IS NOT NULL
```

章の講義項目

- 行の制限に使用する機能
 - WHERE句
 - =、<=、BETWEEN、IN、LIKE、NULL演算子を使用した比較条件
 - AND、ORおよびNOT演算子を使用した論理条件
- 式内の演算子の優先順位の規則
- ORDER BY句を使用した行のソート
- 置換変数
- DEFINEおよびVERIFYコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

優先順位の規則

演算子	意味
1	算術演算子
2	連結演算子
3	比較条件
4	IS [NOT] NULL、LIKE、[NOT] IN
5	[NOT] BETWEEN
6	等しくない
7	NOT論理条件
8	AND論理条件
9	OR論理条件

カッコを使用すると、優先順位の規則を変更できます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

優先順位の規則

優先順位の規則は、式が評価されて計算される順序を決定します。スライドの表は、優先順位のデフォルトの順序を示しています。ただし、最初に計算する式をカッコで囲んで、デフォルトの順序を変更することができます。

優先順位の規則

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR      job_id = 'AD_PREP'
AND      salary > 15000;
```

1

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PREP	24000
2	Abel	SA_REP	11000
3	Taylor	SA_REP	8600
4	Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR      job_id = 'AD_PREP')
AND      salary > 15000;
```

2

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PREP	24000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

優先順位の規則(続き)

1. AND演算子の優先順位: 例

この例では、次の2つの条件があります。

- 最初の条件は、職務IDがAD_PREPであるとともに、給与が\$15,000を超えるということです。
- 2つ目の条件は、職務IDがSA_REPであるということです。

したがって、SELECT文は次のように解釈されます。

“従業員が代表取締役であるとともに給与が\$15,000を超える場合、または従業員が販売担当者である場合に行を選択する。”

2. カッコの使用: 例

この例では、次の2つの条件があります。

- 最初の条件は、職務IDがAD_PREPまたはSA_REPであるということです。
- 2つ目の条件は、給与が\$15,000を超えるということです。

したがって、SELECT文は次のように解釈されます。

“従業員が代表取締役または販売担当者であるとともに、その従業員の給与が\$15,000を超える場合に行を選択する。”

章の講義項目

- 行の制限に使用する機能
 - WHERE句
 - =、<=、BETWEEN、IN、LIKE、NULL演算子を使用した比較条件
 - AND、ORおよびNOT演算子を使用した論理条件
- 式内の演算子の優先順位の規則
- ORDER BY句を使用した行のソート
- 置換変数
- DEFINEおよびVERIFYコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ORDER BY句の使用方法

- ORDER BY句は、次のように取得した行をソートするために使用します。
 - ASC: 昇順(デフォルト)
 - DESC: 降順
- ORDER BY句は、次のようにSELECT文の最後に指定します。

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES		90 17-JUN-87
2	Whalen	AD_ASST		10 17-SEP-87
3	Kochhar	AD_VP		90 21-SEP-89
4	Hunold	IT_PROG		60 03-JAN-90
5	Ernst	IT_PROG		60 21-MAY-91
6	De Haan	AD_VP		90 13-JAN-93

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ORDER BY句の使用方法

問合せ結果として戻される行の順序は定義されていません。このような行のソートにORDER BY句を使用できます。ただし、ORDER BY句を使用する場合は、SQL文の最後の句にする必要があります。ソート条件として、さらに式、別名または列位置を指定できます。

構文

```
SELECT      expr
FROM        table
[WHERE      condition(s)]
[ORDER BY  {column, expr, numeric_position} [ASC|DESC]];
```

構文の内容

ORDER BY 取得した行の表示順序を指定します。
ASC 行を昇順(つまりデフォルトの順序)にソートします。
DESC 行を降順にソートします。

ORDER BY句を使用しない場合はソート順序が未定義になり、Oracleサーバーで同じ問合せを2回行っても行が同じ順序でフェッチされないことがあります。行を特定の順序で表示するには、ORDER BY句を使用してください。

注意: キーワードNULLS FIRSTまたはNULLS LASTを使用して、NULL値が含まれる戻された行が順序の最初にくるか、最後にくるかを指定します。

ソート

- 降順でソート:

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date DESC ;
```

1

- 列の別名を指定してソート:

```
SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal ;
```

2

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ソート

デフォルトのソート順序は昇順で、次のようになります。

- 数値は最も小さい値が最初に表示されます。たとえば、1～999となります。
- 日付値は最も前の値が最初に表示されます。たとえば、01-JAN-92の方が01-JAN-95よりも前に表示されます。
- 文字値はアルファベット順に表示されます。たとえば、“A”が最初に表示されて“Z”が最後に表示されます。
- NULL値は昇順の場合は最後、降順の場合は最初に表示されます。
- SELECTリストにない列に基づいてソートすることもできます。

例:

- 行の表示順序を逆にするには、ORDER BY句で列名の後にDESCキーワードを指定します。スライドの例では、最近雇用された従業員が最初に表示されるように結果がソートされます。
- ORDER BY句で列の別名を使用することもできます。スライドの例では、年間収入によってデータがソートされます。

ソート

- 列の位置を数値で指定してソート:

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY 3;
```

3

- 複数の列を指定してソート:

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC;
```

4

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ソート(続き)

例:

3. SELECT句内での列の位置を数値で指定して、問合せ結果をソートできます。スライドの例では、department_idによって結果がソートされます。これは、この列がSELECT句内で3番目の位置にあるためです。
4. 問合せ結果を、複数の列に基づいてソートできます。このソートが基づく列数の上限は、指定した表内の列数です。ORDER BY句で複数の列を指定し、カンマで列名を区切ります。列の順序を逆にする場合は、その名前の後にDESCを指定します。

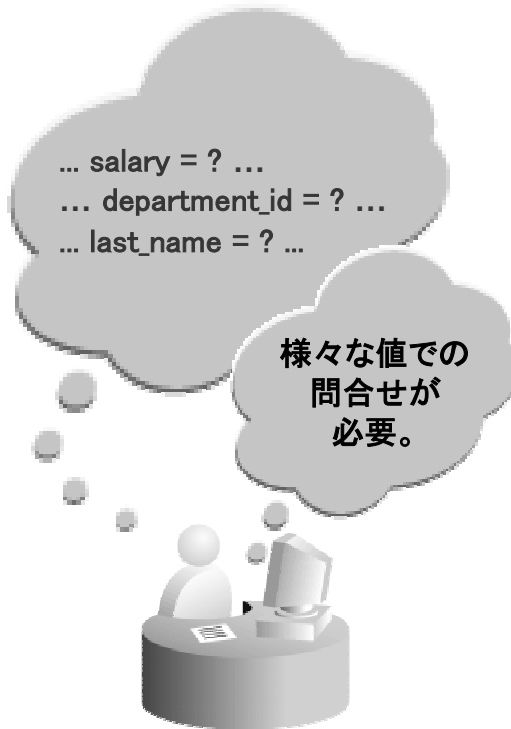
章の講義項目

- 行の制限に使用する機能
 - WHERE句
 - =、<=、BETWEEN、IN、LIKE、NULL演算子を使用した比較条件
 - AND、ORおよびNOT演算子を使用した論理条件
- 式内の演算子の優先順位の規則
- ORDER BY句を使用した行のソート
- 置換変数
- DEFINEおよびVERIFYコマンド

ORACLE

Copyright © 2007, Oracle. All rights reserved.

置換変数



ORACLE

Copyright © 2007, Oracle. All rights reserved.

置換変数

これまでに、すべてのSQL文を、事前に定義された列、条件およびその値に対して実行しました。ここでは、job_IDがSA_REPである従業員だけでなく、様々な職務を持つ従業員リストを表示する問合せが必要であることを想定します。WHERE句を編集して、コマンドを実行するたびに別の値を指定できますが、さらに簡単な方法もあります。

WHERE句で具体的な値の代わりに置換変数を使用すると、同じ問合せを様々な値を使用して実行することができます。

置換変数を使用すると、ユーザーに値の入力を求めて、戻されるデータの範囲を制限するレポートを作成できます。置換変数はコマンド・ファイルまたは単一のSQL文に埋め込むことができます。変数は、値を一時的に格納するコンテナと見なすことができます。SQL文の実行時に、格納されているこの値が使用されます。

置換変数

- 置換変数は、次のように使用します。
 - シングランパサンド(&)およびダブルアンパサンド(&&)置換を使用して、一時的に値を格納する
- 置換変数を使用して次の要素を補完します。
 - WHERE条件
 - ORDER BY句
 - 列の式
 - 表名
 - SELECT文全体

ORACLE

Copyright © 2007, Oracle. All rights reserved.

置換変数(続き)

シングランパサンド(&)置換変数を使用すると、値を一時的に格納できます。

DEFINEコマンドを使用して、変数を事前定義することもできます。DEFINEは値を作成して変数に割り当てます。

データの制限付き範囲: 例

- 現行の四半期または指定されたデータ範囲のみを対象とする数値のレポート
- レポートを要求したユーザーのみに関連するデータのレポート
- 特定の部門のみの人員の表示

その他の対話的な効果

対話的な効果は、WHERE句とのユーザーとの直接の対話に制限されません。同じ原理を次のような他の目的にも使用できます。

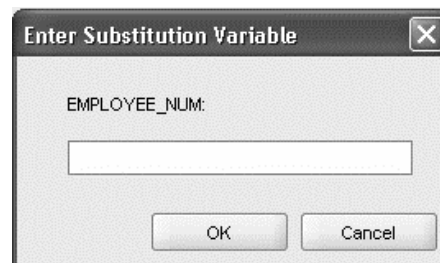
- 人ではなくファイルからの入力値の取得
- あるSQL文から別のSQL文への値の受渡し

注意: SQL DeveloperとSQL* Plusは、どちらも置換変数とDEFINE/UNDEFINEコマンドをサポートしています。ただし、SQL DeveloperまたはSQL* Plusは、ユーザー入力に対する妥当性チェックをサポートしていません(データ型を除く)。

シングルアンパサンド置換変数の 使用方法

先頭にアンパサンド(&)を付けた変数を使用すると、次のようにユーザーに値の入力が求められます。

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```



ORACLE

Copyright © 2007, Oracle. All rights reserved.

シングルアンパサンド置換変数の使用方法

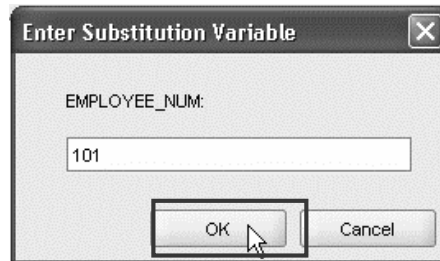
通常、ユーザーは、レポートの実行時に動的に戻されるデータを制限する必要があります。SQL*PlusまたはSQL Developerでは、ユーザー変数を使用することでこのような柔軟性を実現しています。アンパサンド(&)を使用して、SQL文内で各変数を識別します。ただし、各変数の値を定義する必要はありません。

表記法	説明
<i>&user_variable</i>	SQL文内の変数を示します。変数が存在しない場合は、SQL*PlusまたはSQL Developerによって、ユーザーは値を入力するように求められます。新しい変数は使用後に破棄されます。

スライドの例では、従業員番号のSQL Developer置換変数を作成しています。このSQL文が実行されると、SQL Developer によって、ユーザーに従業員番号の入力を求めるプロンプトが表示され、その従業員の従業員番号、姓、給与および部門番号が表示されます。

シングルアンパサンドの場合、変数が存在しなければ、このコマンドを実行するたびにユーザーは入力を求められます。

シングルアンパサンド置換変数の 使用方法



	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

ORACLE

Copyright © 2007, Oracle. All rights reserved.

シングルアンパサンド置換変数の使用方法(続き)

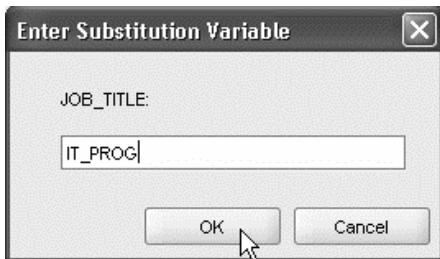
SQL Developerによって、SQL文にアンパサンドが含まれていることが検出されると、SQL文で指定されている置換変数の値を入力するように求められます。

値を入力して「OK」ボタンをクリックすると、結果がSQL Developerセッションの「Results」タブに表示されます。

置換変数での文字値および日付値の使用

日付値および文字値については、次のように一重引用符を使用します。

```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```



The dialog box titled "Enter Substitution Variable" has a close button (X) in the top right corner. It contains a label "JOB_TITLE:" followed by a text input field containing the value "IT_PROG". At the bottom, there are "OK" and "Cancel" buttons. A mouse cursor is pointing at the "OK" button.

	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

ORACLE

Copyright © 2007, Oracle. All rights reserved.

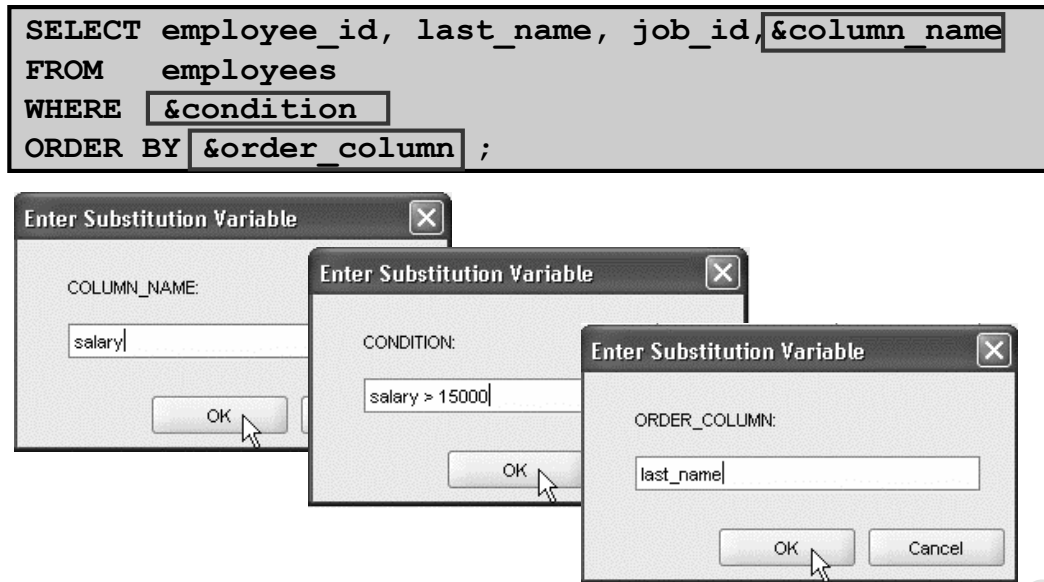
置換変数での文字値および日付値の使用

WHERE句では、日付値および文字値を一重引用符で囲む必要があります。置換変数にも同じ規則が適用されます。

SQL文自体の中で、変数を一重引用符で囲みます。

スライドは、SQL Developer置換変数の職務の値に基づいて、すべての従業員の従業員名、部門番号および年間収入を取得する問合せを示しています。

列名、式およびテキストの指定



Copyright © 2007, Oracle. All rights reserved.

列名、式およびテキストの指定

置換変数はSQL文のWHERE句内だけでなく、列名、式またはテキストの置換用としても使用できます。

例:

スライドの例では、従業員番号、姓、職務、および実行時にユーザーが指定するその他の列が、EMPLOYEES表から表示されます。SELECT文内の置換変数ごとに値を入力するように求められ、「OK」をクリックして続行します。

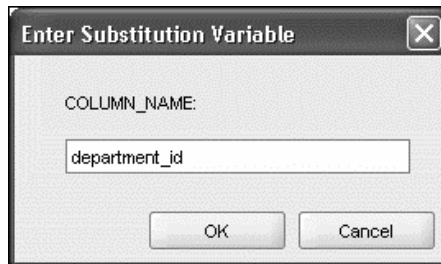
置換変数の値を入力しないと、先行する文の実行時にエラー・メッセージが表示されます。

注意: 置換変数はSELECT文内の任意の場所で使用できますが、コマンド・プロンプトで入力する最初の語を除きます。

ダブルアンパサンド置換変数の使用方法

毎回ユーザーに入力を求めずに変数値を再利用する場合は、次のようにダブルアンパサンド(&&)を使用します。

```
SELECT  employee_id, last_name, job_id, &&column_name
FROM    employees
ORDER BY &column_name ;
```



A dialog box titled "Enter Substitution Variable" with a close button (X). It contains a label "COLUMN_NAME:" and a text input field with the value "department_id". At the bottom are "OK" and "Cancel" buttons.

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ダブルアンパサンド置換変数の使用方法

毎回ユーザーに値の入力を指示せずに変数値を再利用する場合は、ダブルアンパサンド(&&)置換変数を使用できます。ユーザーに対しては、値の入力が1回だけ求められます。スライドの例の場合、ユーザーは変数column_nameの値を指定するように1回だけ指示されます。ユーザーによって指定された値(department_id)は、データの表示と順序付けの両方に使用されます。この問合せを再度実行した場合、その変数の値の入力を指示されることはありません。

SQL Developerでは、DEFINEコマンドを使用して指定された値が格納されます。変数名を参照するたびにその値が再利用されます。ユーザー変数を指定した後は、次のようにUNDEFINEコマンドを使用して削除する必要があります。

```
UNDEFINE column_name
```

章の講義項目

- 行の制限に使用する機能
 - WHERE句
 - =、<=、BETWEEN、IN、LIKE、NULL演算子を使用した比較条件
 - AND、ORおよびNOT演算子を使用した論理条件
- 式内の演算子の優先順位の規則
- ORDER BY句を使用した行のソート
- 置換変数
- DEFINEおよびVERIFYコマンド

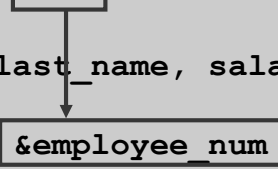
ORACLE

Copyright © 2007, Oracle. All rights reserved.

DEFINEコマンドの使用方法

- DEFINEコマンドは、値を作成し変数に割り当てるために使用します。
- UNDEFINEコマンドは、変数を削除するために使用します。

```
DEFINE employee_num = 200  
  
SELECT employee_id, last_name, salary, department_id  
FROM employees  
WHERE employee_id = &employee_num ;  
  
UNDEFINE employee_num
```



ORACLE

Copyright © 2007, Oracle. All rights reserved.

DEFINEコマンドの使用方法

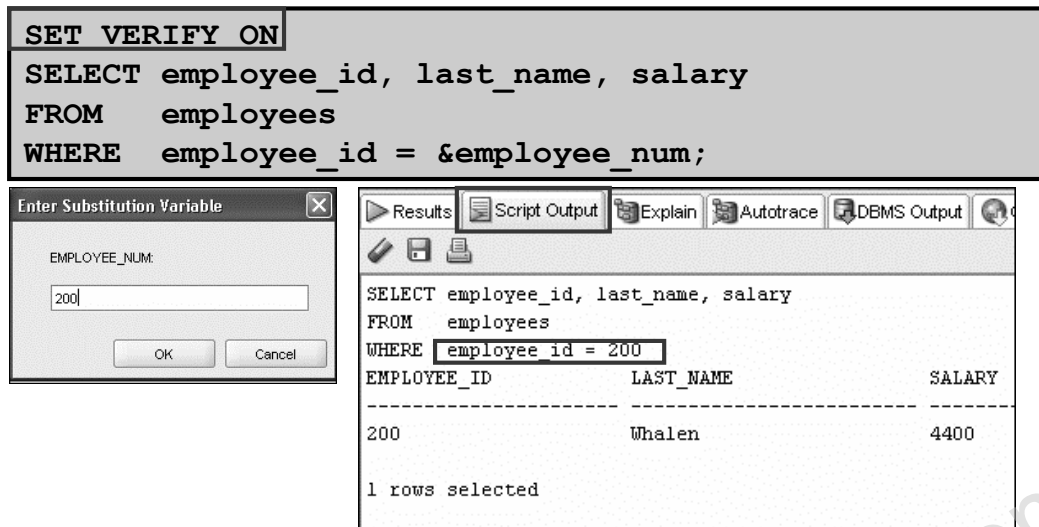
この例では、DEFINEコマンドを使用して従業員番号の置換変数を作成しています。実行時には、その従業員の従業員番号、名前、給与および部門番号が表示されます。

この置換変数はSQL DeveloperのDEFINEコマンドを使用して作成されているため、ユーザーは従業員番号の値を入力するように求められません。その代わりに、SELECT文内で自動的に定義済の変数値に置換えられます。

EMPLOYEE_NUM置換変数は、ユーザーがその定義を解除するか、SQL Developerセッションを終了するまでセッションに存在します。

VERIFYコマンドの使用方法

VERIFYコマンドは、SQL Developerによって置換変数が値で置換えられる前および後に、置換変数の表示を切り替えるために使用します。



Copyright © 2007, Oracle. All rights reserved.

VERIFYコマンドの使用方法

SQL文内の変更を確認するには、VERIFYコマンドを使用します。SET VERIFY ONを設定すると、置換変数を値で置換えた後にSQL Developerによって強制的にコマンドのテキストが表示されます。VERIFY出力を表示するには、SQL Worksheetで「Run Script」アイコン ([F5]キー)を使用する必要があります。SQL Developerによって置換変数が値で置換えられた後に、スライドに示すようにコマンドのテキストが「Script Output」タブに表示されます。

スライドの例では、SQL文のEMPLOYEE_ID列の新しい値が表示されてから、出力が表示されています。

SQL*Plusシステム変数

SQL*Plusでは、作業環境を制御する様々なシステム変数を使用します。そのような変数の1つとしてVERIFYがあります。すべてのシステム変数のリストを表示するには、SQL*Plusコマンド・プロンプトでSHOW ALLコマンドを実行します。

まとめ

この章では、次のことを学習しました。

- WHERE句を使用した出力の行の制限:
 - 比較条件を使用する
 - BETWEEN、IN、LIKEおよびNULL演算子を使用する
 - AND、ORおよびNOT論理演算子を適用する
- ORDER BY句を使用した出力の行のソート:

```
SELECT  *|{[DISTINCT] column|expression [alias],...}  
FROM    table  
[WHERE  condition(s)]  
[ORDER BY {column, expr, alias} [ASC|DESC]] ;
```

- アンパサンド置換を使用した実行時の出力の制限とソート

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

この章では、SELECT文によって戻される行の制限とソートについて学習しました。また、様々な演算子と条件を実装する方法についても学習しました。

置換変数を使用することにより、SQL文に柔軟性を持たせることができます。この方法により、問合せで実行時に行のフィルタ条件の指定を求めることができます。

演習2: 概要

この演習では次の項目について説明しています。

- データの選択と表示される行の順序の変更
- WHERE句を使用した行の制限
- ORDER BY句を使用した行のソート
- 置換変数を使用したSQL SELECT文への柔軟性の追加

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習2: 概要

この演習では、WHERE句とORDER BY句を使用する文を含めて、さらにレポートを作成します。アンパサンド置換を含めることで、SQL文の再利用性と汎用性が向上します。

演習2

HR部門から、いくつかの問合せの作成支援を依頼されました。

1. HR部門では、予算上の問題で、給与が\$12,000を超える従業員の姓と給与を表示するレポートを必要としています。作成したSQL文は、lab_02_01.sqlという名前のファイルとして保存します。この問合せを実行してください。

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000

2. 新しいSQL Worksheetを開きます。従業員番号176の姓と部門番号を表示するレポートを作成します。この問合せを実行してください。

	LAST_NAME	DEPARTMENT_ID
1	Taylor	80

3. HR部門は、給与の高い従業員と低い従業員を検索する必要があります。lab_02_01.sqlを変更して、給与が\$5,000～\$12,000の範囲にない従業員の姓と給与を表示するようにします。作成したSQL文を、lab_02_03.sqlとして保存してください。

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Lorentz	4200
5	Rajs	3500
6	Davies	3100
7	Matos	2600
8	Vargas	2500
9	Whalen	4400
10	Hartstein	13000

4. 姓がMatosおよびTaylorである従業員の姓、職務IDおよび開始日を表示するレポートを作成します。開始日に基づいて、問合せを昇順でソートしてください。

	LAST_NAME	JOB_ID	HIRE_DATE
1	Matos	ST_CLERK	15-MAR-98
2	Taylor	SA_REP	24-MAR-98

演習2(続き)

5. 部門20または50に所属するすべての従業員の姓と部門番号を、名前のアルファベット順(昇順)で表示します。

	LAST_NAME	DEPARTMENT_ID
1	Davies	50
2	Fay	20
3	Hartstein	20
4	Matos	50
5	Mourgos	50
6	Rajs	50
7	Vargas	50

6. lab_02_03.sqlを変更して、給与が\$5,000～\$12,000の範囲にあり部門20または50に所属している従業員の姓と給与を表示するようにします。列にそれぞれEmployeeおよびMonthly Salaryというラベルを付けます。lab_02_03.sqlは、lab_02_06.sqlとして保存し直します。lab_02_06.sqlの文を実行してください。

	Employee	Monthly Salary
1	Fay	6000
2	Mourgos	5800

7. HR部門は、1994年に雇用されたすべての従業員の姓と雇用日を表示するレポートを必要としています。

	LAST_NAME	HIRE_DATE
1	Higgins	07-JUN-94
2	Gietz	07-JUN-94

8. 担当マネージャがいないすべての従業員の姓と職務を表示するレポートを作成します。

	LAST_NAME	JOB_ID
1	King	AD_PRES

9. 歩合を受け取るすべての従業員の姓、給与および歩合を表示するレポートを作成します。データを、給与および歩合の降順でソートします。ORDER BY句で、列の位置を数値で指定してください。

	LAST_NAME	SALARY	COMMISSION_PCT
1	Abel	11000	0.3
2	Zlotkey	10500	0.2
3	Taylor	8600	0.2
4	Grant	7000	0.15

演習2(続き)

10. HR部門のメンバーは、作成中の問合せにさらに柔軟性を求めています。求められる柔軟性とは、レポートで、プロンプト表示後にユーザーが指定する額を超える給与の従業員の姓と給与が表示されるようにすることです。この問合せをlab_02_10.sqlという名前のファイルに保存してください。プロンプトが表示されたときに12000を入力すると、レポートに次の結果が表示されます。

	R Z	LAST_NAME	R Z	SALARY
1		King		24000
2		Kochhar		17000
3		De Haan		17000
4		Hartstein		13000

11. HR部門では、マネージャに基づいたレポートを実行する必要があります。プロンプトを表示してユーザーにマネージャIDの入力を求め、そのマネージャが管理する従業員の従業員ID、姓、給与および部門を生成する問合せを作成します。HR部門は、選択した列でレポートをソートする機能を必要としています。データのテストには、次の値を使用できます。

manager_id = 103、last_nameでソート:

	R Z	EMPLOYEE_ID	R Z	LAST_NAME	R Z	SALARY	R Z	DEPARTMENT_ID
1		104		Ernst		6000		60
2		107		Lorentz		4200		60

manager_id = 201、salaryでソート:

	R Z	EMPLOYEE_ID	R Z	LAST_NAME	R Z	SALARY	R Z	DEPARTMENT_ID
1		202		Fay		6000		20

manager_id = 124、employee_idでソート:

	R Z	EMPLOYEE_ID	R Z	LAST_NAME	R Z	SALARY	R Z	DEPARTMENT_ID
1		141		Rajs		3500		50
2		142		Davies		3100		50
3		143		Matos		2600		50
4		144		Vargas		2500		50

演習2(続き)

時間があるときは、次の演習問題に進みます。

12. 姓の3文字目が“a”であるすべての従業員の姓を表示します。

	LAST_NAME
1	Grant
2	Whalen

13. 姓の中に“a”と“e”の両方が含まれるすべての従業員の姓を表示します。

	LAST_NAME
1	Davies
2	De Haan
3	Hartstein
4	Whalen

さらに演習を続ける場合は、次の演習問題に進みます。

14. 職務が販売担当者または在庫管理者であり、給与が\$2,500、\$3,500または\$7,000と等しくないすべての従業員の姓、職務および給与を表示します。

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000
2	Taylor	SA_REP	8600
3	Davies	ST_CLERK	3100
4	Matos	ST_CLERK	2600

15. lab_02_06.sqlを変更して、歩合が20%であるすべての従業員の姓、給与および歩合を表示するようにします。lab_02_06.sqlをlab_02_15.sqlとして保存し直します。lab_02_15.sqlの文を再実行してください。

	Employee	Monthly Salary	COMMISSION_PCT
1	Zlotkey	10500	0.2
2	Taylor	8600	0.2

単一行関数を使用した出力のカスタマイズ

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

目的

この章を終えると、次のことができるようになります。

- SQLで利用できる各種関数の説明
- SELECT文での文字、数値および日付関数の使用

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

関数は基本的な問合せブロックをより強力にし、データ値を操作するために使用されます。この章は、関数について説明する2つの章のうちの最初の章です。単一行の文字、数値および日付関数を中心に説明します。

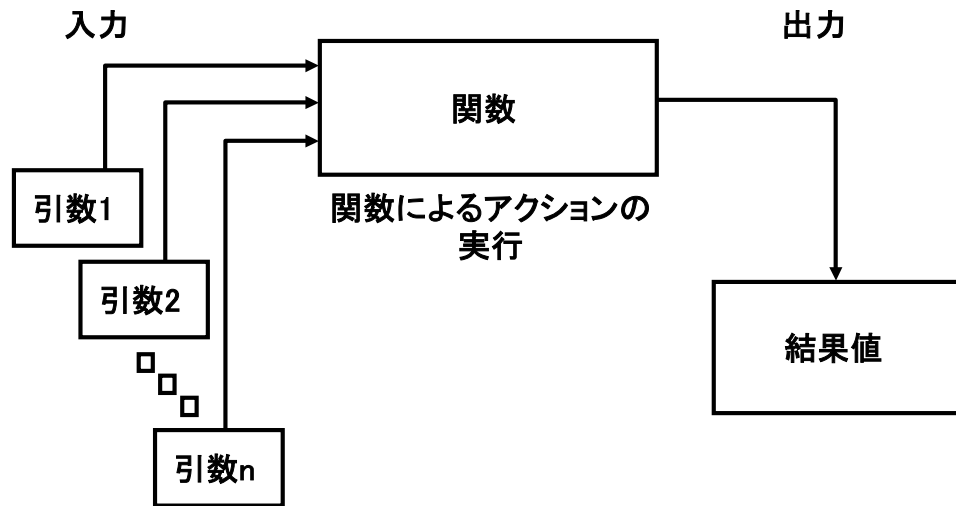
章の講義項目

- 単一行SQL関数
- 文字関数
- 数値関数
- 日付の操作
- 日付関数

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL関数



ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL関数

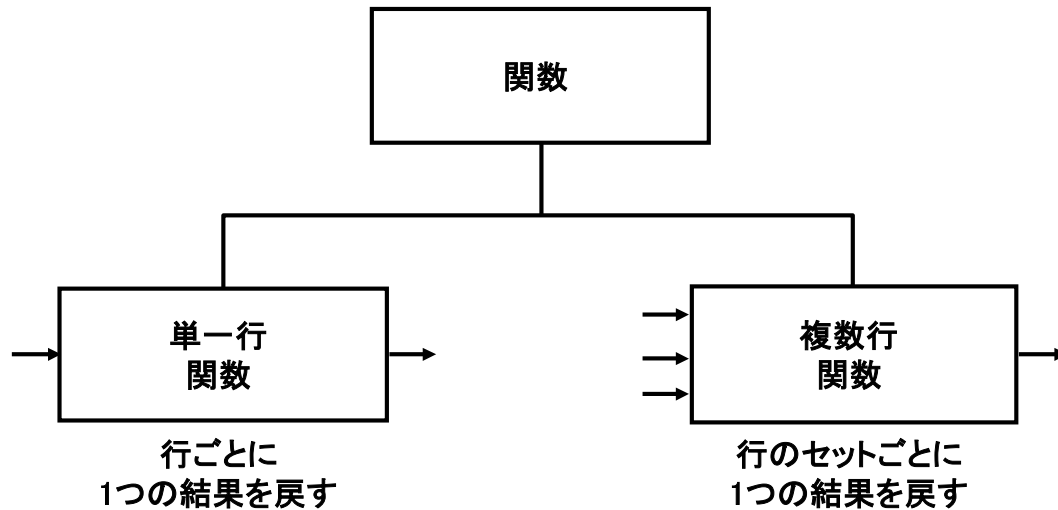
関数は非常に強力なSQL機能です。関数を使用して、次のことを実行できます。

- データに対する計算の実行
- それぞれのデータ項目の変更
- 行のグループに対する出力の操作
- 表示する日付と数値の書式設定
- 列のデータ型の変換

SQL関数は引数を使用することがあり、常に値を戻します。

注意: 関数がSQL:2003準拠の関数かどうかを確認するには、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のCore SQL:2003に対するOracleの準拠に関する節を参照してください。

SQL関数の2つのタイプ



ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL関数の2つのタイプ

関数には、次の2つのタイプがあります。

- 単一行関数
- 複数行関数

単一行関数

この関数は単一の行でのみ動作し、行ごとに1つの結果を返します。様々なタイプの単一行関数があります。この章では、次のタイプの関数について説明します。

- 文字
- 数値
- 日付
- 変換
- 汎用

複数行関数

関数は、行のグループを操作し、行のグループごとに1つの結果を返すことができます。この関数は、グループ関数(第5章「グループ関数を使用した集計データのレポート」を参照)とも呼ばれます。

注意: 使用可能な関数とその構文の詳細および完全なリストは、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』の関数に関するトピックを参照してください。

単一行関数

単一行関数の機能:

- データ項目を操作します。
- 引数を受け入れて、1つの値を返します。
- 戻される各行に対して動作します。
- 行ごとに1つの結果を返します。
- データ型を変更できます。
- ネストできます。
- 列または式の形式の引数を受け入れます。

```
function_name [(arg1, arg2,...)]
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

単一行関数

単一行関数は、データ項目を操作する場合に使用します。単一行関数は1つ以上の引数を受け入れて、問合せによって戻される行ごとに1つの値を返します。引数には次のいずれかを使用できます。

- ユーザー指定の定数
- 変数値
- 列名
- 式

単一行関数には次の機能があります。

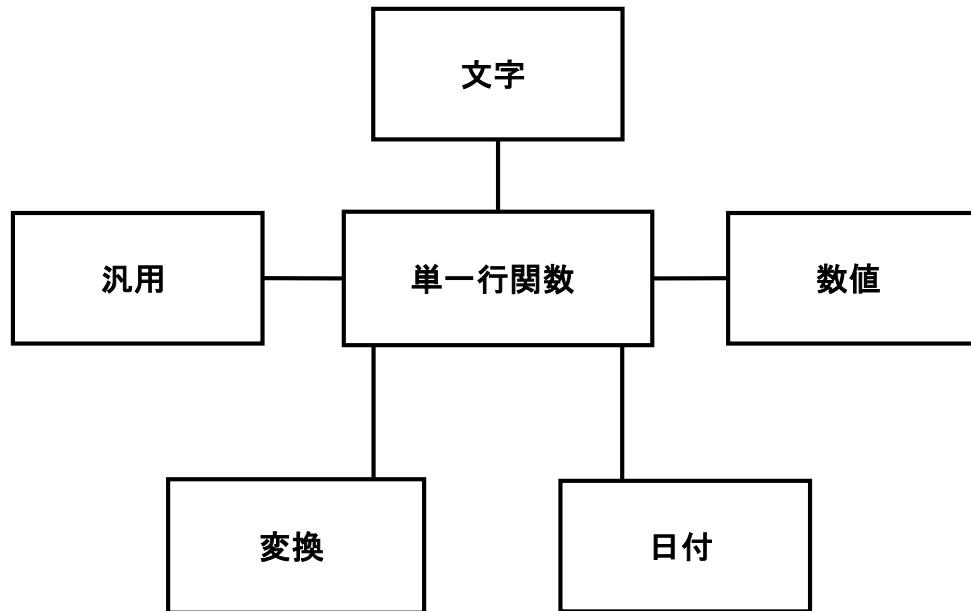
- 問合せで戻される各行に対して動作します。
- 行ごとに1つの結果を返します。
- 参照される型とは異なる型のデータ値を返すことができます。
- 1つ以上の引数を使用できます。
- SELECT、WHEREおよびORDER BY句で使用できます。ネストすることもできます。

構文の内容:

function_name 関数の名前です。

arg1, arg2 関数によって使用される任意の引数です。列名または式を指定できます。

単一行関数



ORACLE

Copyright © 2007, Oracle. All rights reserved.

単一行関数(続き)

この章では次の単一行関数について説明します。

- **文字関数:** 文字入力を受け入れて、文字値と数値の両方を戻すことができます。
- **数値関数:** 数値入力を受け入れて、数値を戻すことができます。
- **日付関数:** DATEデータ型の値に対して動作します(数値を戻すMONTHS_BETWEEN関数を除くすべての日付関数は、DATEデータ型の値を戻します)。

次の単一行関数については、次の章の「変換関数と条件式の使用方法」で説明します。

- **変換関数:** 値を1つのデータ型から別のデータ型に変換します。
- **汎用関数:**
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
 - CASE
 - DECODE

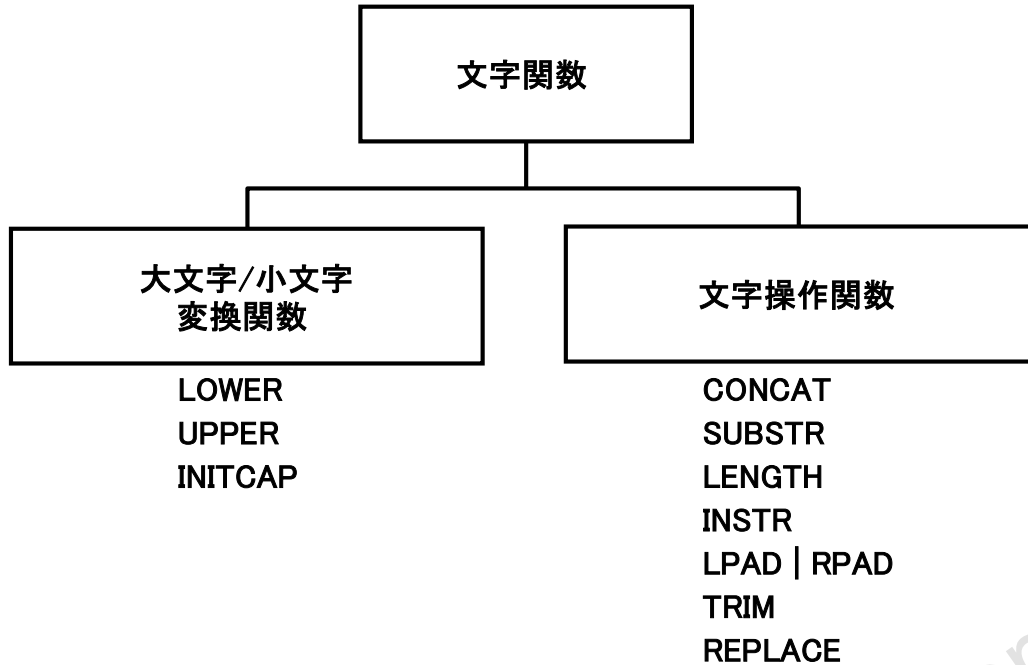
章の講義項目

- 単一行SQL関数
- 文字関数
- 数値関数
- 日付の操作
- 日付関数

ORACLE

Copyright © 2007, Oracle. All rights reserved.

文字関数



ORACLE

Copyright © 2007, Oracle. All rights reserved.

文字関数

単一行文字関数は、文字データを入力として受け入れて、文字値と数値の両方を戻すことができます。文字関数は、次のように分類できます。

- 大文字/小文字変換関数
- 文字操作関数

関数	目的
LOWER(<i>column/expression</i>)	アルファベットの文字値を小文字に変換します。
UPPER(<i>column/expression</i>)	アルファベットの文字値を大文字に変換します。
INITCAP(<i>column/expression</i>)	それぞれの語の最初の文字に対してアルファベットの文字値を大文字に変換します。他のすべての文字は小文字になります。
CONCAT(<i>column1/expression1</i> , <i>column2/expression2</i>)	最初の文字値と2番目の文字値を連結します。連結演算子()と同等です。
SUBSTR(<i>column/expression</i> , <i>m</i> [, <i>n</i>])	文字位置 <i>m</i> を起点とする <i>n</i> 文字長の文字値から指定された文字を戻します(<i>m</i> が負の場合は、文字値の末尾から数えます。 <i>n</i> を省略すると、文字列の末尾までのすべての文字が戻されます)。

注意: この章では、使用可能な一部の関数についてのみ説明します。

文字関数(続き)

関数	目的
LENGTH(<i>column</i> / <i>expression</i>)	式内の文字数を返します。
INSTR(<i>column</i> / <i>expression</i> , ' <i>string</i> ', [, <i>m</i>], [<i>n</i>])	名前付きの文字列の位置を数値で返します。必要に応じて、検索を開始する位置 <i>m</i> および文字列の出現箇所 <i>n</i> を指定できます。 <i>m</i> および <i>n</i> のデフォルトは1です。この場合、文字列の先頭から検索が開始されて、最初に出現した箇所がレポートされます。
LPAD(<i>column</i> / <i>expression</i> , <i>n</i> , ' <i>string</i> ') RPAD(<i>column</i> / <i>expression</i> , <i>n</i> , ' <i>string</i> ')	文字式で、文字の長さが <i>n</i> になるまで左側に文字を埋め込んだ式を返します。 文字式で、文字の長さが <i>n</i> になるまで右側に文字を埋め込んだ式を返します。
TRIM(<i>leading</i> / <i>trailing</i> / <i>both</i> , <i>trim_character</i> FROM <i>trim_source</i>)	文字列から先行文字または後続文字(あるいはその両方)を切り捨てます。 <i>trim_character</i> または <i>trim_source</i> が文字リテラルの場合は、一重引用符で囲む必要があります。これはOracle8i以降のバージョンで利用できる機能です。
REPLACE(<i>text</i> , <i>search_string</i> , <i>replacement_string</i>)	文字列のテキスト式を検索し、検出した場合は、指定された置換文字列に置換します。

注意: SQL:2003に完全または部分的に準拠している関数の一部を次に示します。

UPPER

LOWER

TRIM

LENGTH

SUBSTR

INSTR

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のCore SQL:2003に対するOracleの準拠に関する節を参照してください。

大文字/小文字変換関数

次の関数は、文字列の大文字と小文字を変換します。

関数	結果
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

ORACLE

Copyright © 2007, Oracle. All rights reserved.

大文字/小文字変換関数

LOWER、UPPERおよびINITCAPという3つの大文字/小文字変換関数があります。

- LOWER: 大文字と小文字が混在する文字列または大文字の文字列を小文字に変換します。
- UPPER: 大文字と小文字が混在する文字列または小文字の文字列を大文字に変換します。
- INITCAP: それぞれの語の最初の文字を大文字に変換し、他のすべての文字を小文字にします。

```
SELECT 'The job id for '||UPPER(last_name)||' is '  
      ||LOWER(job_id) AS "EMPLOYEE DETAILS"  
FROM   employees;
```

	EMPLOYEE DETAILS
1	The job id for ABEL is sa_rep
2	The job id for DAVIES is st_clerk
3	The job id for DE HAAN is ad_vp
...	
19	The job id for WHALEN is ad_asst
20	The job id for ZLOTKEY is sa_man

大文字/小文字変換関数の使用方法

従業員Higginsの従業員番号、姓および部門番号は、次のように表示します。

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE last_name = 'higgins';
```

0 rows selected

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LOWER(last_name) = 'higgins';
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205	Higgins	110

ORACLE

Copyright © 2007, Oracle. All rights reserved.

大文字/小文字変換関数の使用方法

スライドの例では、従業員Higginsの従業員番号、姓および部門番号が表示されています。

最初のSQL文のWHERE句は、従業員名をhigginsとして指定しています。EMPLOYEES表のすべてのデータは大文字と小文字を区別して格納されているため、名前higginsは表内の検索で一致しません。したがって、行は選択されません。

2番目のSQL文のWHERE句は、higginsと比較されるEMPLOYEES内の従業員名を指定し、比較するためにLAST_NAME列を小文字に変換します。これで、いずれの名前も小文字になるので、一致が検出されて1つの行が選択されます。WHERE句を次のように書き直しても、同じ結果を得ることができます。

```
...WHERE last_name = 'Higgins'
```

出力では、データベースに格納されていたとおりに名前が表示されます。名前を大文字で表示するには、SELECT文でUPPER関数を使用します。

```
SELECT employee_id, UPPER(last_name), department_id
FROM employees
WHERE INITCAP(last_name) = 'Higgins';
```

文字操作関数

次の関数は文字列を操作します。

関数	結果
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ('JACK and JUE', 'J', 'BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

ORACLE

Copyright © 2007, Oracle. All rights reserved.

文字操作関数

この章では、CONCAT、SUBSTR、LENGTH、INSTR、LPAD、RPADおよびTRIMの文字操作関数について説明します。

- CONCAT: 値を結合します (CONCATでは2つまでパラメータを使用できます)。
- SUBSTR: 指定された長さの文字列を抽出します。
- LENGTH: 文字列の長さを数値で表示します。
- INSTR: 名前付きの文字の位置を示す数値を検索します。
- LPAD: 文字式で、文字の長さが *n* になるまで左側に文字を埋め込んだ式を戻します。
- RPAD: 文字式で、文字の長さが *n* になるまで右側に文字を埋め込んだ式を戻します。
- TRIM: 文字列から先行文字または後続文字 (あるいはその両方) を切り捨てます (*trim_character* または *trim_source* が文字リテラルの場合は、一重引用符で囲む必要があります)。

注意: UPPERやLOWERなどの関数をアンパサンド置換と一緒に使用できます。たとえば、UPPER('&job_title')を使用すると、大文字または小文字を区別せずに職務を入力できます。

文字操作関数の使用方法

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';

```

① ② ③

	EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
1	174	EllenAbel	SA_REP	4	0
2	176	JonathonTaylor	SA_REP	6	2
3	178	KimberelyGrant	SA_REP	5	3
4	202	PatFay	MK_REP	3	2

① ② ③

ORACLE

Copyright © 2007, Oracle. All rights reserved.

文字操作関数の使用方法

スライドの例では、職務IDの4番目の位置を起点にして文字列REPが職務IDに含まれているすべての従業員に対して、連結された従業員の姓と名前、従業員の姓の長さ、および従業員の姓に含まれる文字「a」の位置を示す数値が表示されています。

例:

姓の終わりに文字「n」が付く従業員のデータを表示するようにスライド内のSQL文を変更します。

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';

```

	EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
1	102	LexDe Haan	7	5
2	200	JenniferWhalen	6	3
3	201	MichaelHartstein	9	2

章の講義項目

- 単一行SQL関数
- 文字関数
- **数値関数**
- 日付の操作
- 日付関数

ORACLE

Copyright © 2007, Oracle. All rights reserved.

数値関数

- ROUND: 指定された小数位に値を丸めます。
- TRUNC: 指定された小数位に値を切り捨てます。
- MOD: 除算の余りを戻します。

関数	結果
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100

ORACLE

Copyright © 2007, Oracle. All rights reserved.

数値関数

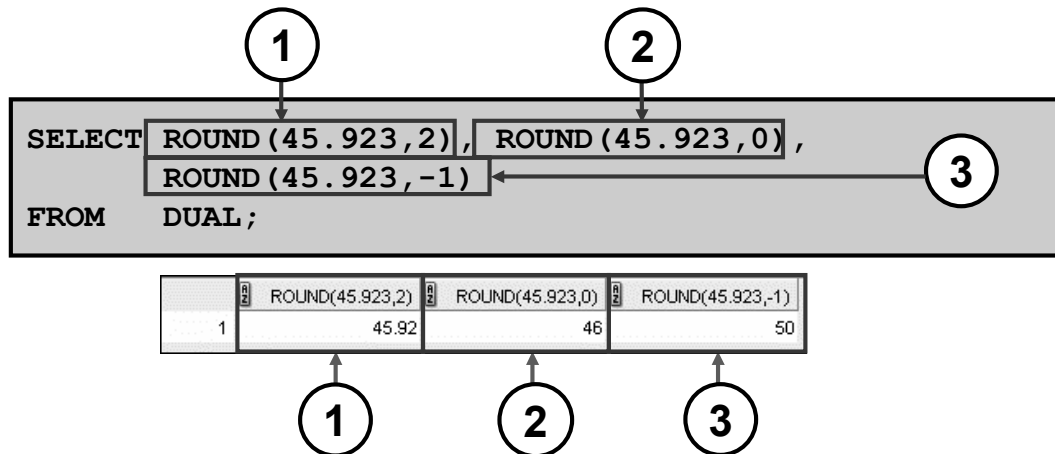
数値関数は数値入力を受け入れて、数値を戻します。この節では、一部の数値関数について説明します。

関数	目的
ROUND(<i>column</i> <i>expression</i> , <i>n</i>)	列、式または値を小数第 <i>n</i> 位に丸めます。 <i>n</i> が省略されている場合は、小数位を使用しません <i>n</i> が負の場合は、小数点の左桁の数値を丸めます。
TRUNC(<i>column</i> <i>expression</i> , <i>n</i>)	列、式または値を小数第 <i>n</i> 位に切り捨てます。 <i>n</i> が省略されている場合は、デフォルトで <i>n</i> はゼロに設定されます。
MOD(<i>m</i> , <i>n</i>)	<i>m</i> を <i>n</i> で除算したときの余りを戻します。

注意: このリストには、使用可能な一部の数値関数のみが含まれます。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』の数値関数に関する節を参照してください。

ROUND関数の使用方法



DUALは、関数および計算の結果を表示するときに使用できるダミー表です。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ROUND関数の使用方法

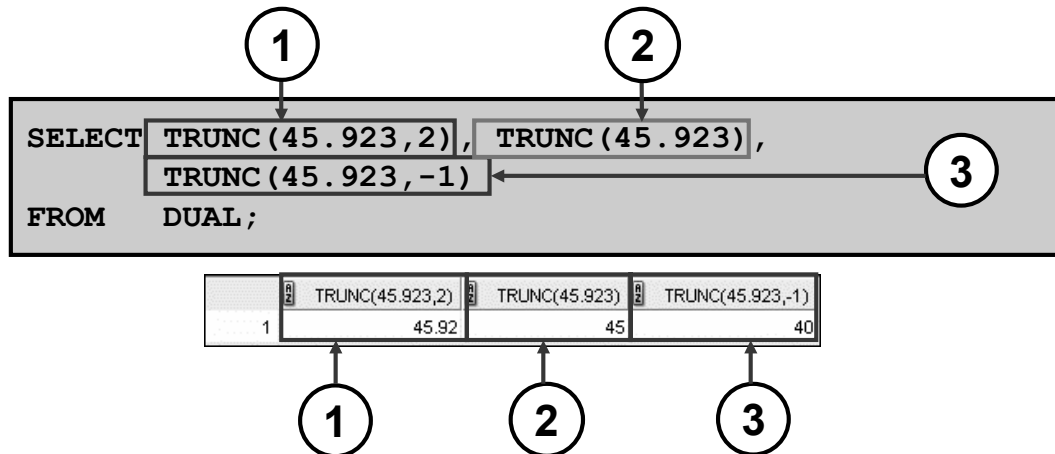
ROUND関数は、列、式または値を小数第 n 位に丸めます。2番目の引数が0の場合または省略されている場合、値は小数位のない状態に丸められます。2番目の引数が2の場合、値は小数第2位に丸められます。また、2番目の引数が-2の場合、値は小数点の左2桁に丸められます(最も近い100の単位に丸められます)。

ROUND関数は日付関数と一緒に使用することもできます。例については、この章で後述します。

DUAL表

DUAL表は、ユーザーSYSによって所有され、すべてのユーザーがアクセスできます。この表には、DUMMYという1つの列と、X値を持つ1つの行が含まれます。DUAL表は、値を1回だけ戻す場合に便利です(ユーザー・データが存在する表からは導出されない、定数、疑似列または式の値など)。通常、DUAL表は、SELECT句構文を補完するために使用されます。これは、SELECT句とFROM句は両方とも指定する必要がありますが、一部の計算では実際の表からの選択が必要ないためです。

TRUNC関数の使用方法



ORACLE

Copyright © 2007, Oracle. All rights reserved.

TRUNC関数の使用方法

TRUNC関数は、列、式または値を小数第 n 位に切り捨てます。

TRUNC関数では、ROUND関数の場合と同じように引数を使用されます。2番目の引数が0の場合または省略されている場合、値は小数位のない状態に切り捨てられます。2番目の引数が2の場合、値は小数第2位に切り捨てられます。また、2番目の引数が-2の場合、値は小数点の左2桁に切り捨てられます。2番目の引数が-1の場合、値は小数点の左1桁に切り捨てられます。

ROUND関数と同じように、TRUNC関数は日付関数と一緒に使用できます。

MOD関数の使用方法

職務が販売担当者であるすべての従業員に対して、給与を5,000で除算した後の余りを計算します。

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

	LAST_NAME	SALARY	MOD(SALARY,5000)
1	Abel	11000	1000
2	Taylor	8600	3600
3	Grant	7000	2000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

MOD関数の使用方法

MOD関数では、1番目の引数を2番目の引数で除算した場合の余りが求められます。スライドの例では、職務IDがSA_REPであるすべての従業員に対して、給与を5,000で除算した後の余りが計算されています。

注意: MOD関数は、多くの場合、値が奇数か偶数かを確認するために使用されます。

章の講義項目

- 単一行SQL関数
- 文字関数
- 数値関数
- 日付の操作
- 日付関数

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付の操作

- Oracle Databaseでは、世紀、年、月、日、時、分および秒を表す内部的な数値書式で日付が格納されます。
- デフォルトの日付の表示書式は、DD-MON-RRです。
 - 年の最後の2桁だけを指定して、21世紀の日付を20世紀に格納できる
 - 同じ方法で、20世紀の日付を21世紀に格納できる

```
SELECT last_name, hire_date
FROM employees
' ';WHERE hire_date < '01-FEB-88';
```

	LAST_NAME	HIRE_DATE
1	King	17-JUN-87
2	Whalen	17-SEP-87

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付の操作

Oracle Databaseでは、世紀、年、月、日、時、分および秒を表す内部的な数値書式で日付が格納されます。

日付のデフォルトの表示書式および入力書式は、DD-MON-RRです。Oracleの有効な日付は、January 1, 4712 B.C.～December 31, 9999 A.Dです。

スライドの例では、HIRE_DATE列の出力がデフォルトの書式DD-MON-RRで表示されています。ただし、日付はこの書式ではデータベースに格納されません。日時のすべての要素が格納されます。したがって、17-JUN-87などのHIRE_DATEは日、月および年として表示されますが、日付に関連する時刻および世紀の情報も存在します。たとえば、完全なデータは、June 17, 1987, 5:10:43 PMのようになります。

RR日付書式

現在の年	指定された日付	RR書式	YY書式
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		指定された年の2桁	
		0-49	50-99
現在の年の 2桁	0-49	戻される日付は現在の 世紀	戻される日付は現在の 世紀の前の世紀
	50-99	戻される日付は現在の 世紀の後の世紀	戻される日付は現在の 世紀

ORACLE

Copyright © 2007, Oracle. All rights reserved.

RR日付書式

RR日付書式はYY要素と似ていますが、この書式を使用すると異なる世紀を指定できます。YYではなくRR日付書式の要素を使用すると、指定された年の2桁と現在の年の下2桁に応じて戻される世紀の値が変わります。スライドの表に、RR要素の動作の概要を示します。

現在の年	指定された日付	解析後 (RR)	解析後 (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017

Oracle日付書式

このデータは、内部的には次のように格納されます。

CENTURY	YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
19	87	06	17	17	10	43

世紀と2000年

日付列を持つレコードが表に挿入されると、世紀情報がSYSDATE関数から取得されます。ただし、日付列は画面に表示されますが、世紀の要素は表示されません(デフォルト設定)。

DATEデータ型では、常に、4桁の数値(世紀を表す2桁と年を表す2桁)として年情報が内部的に格納されます。たとえば、Oracle Databaseでは、87または04ではなく、1987または2004として年が格納されます。

Oracle Internal & Oracle Academy
Use Only

SYSDATE関数の使用方法

SYSDATEは次の値を戻す関数です。

- 日付
- 時刻

```
SELECT sysdate  
;FROM dual;
```

SYSDATE
1 31-MAY-07

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SYSDATE関数の使用方法

SYSDATEは、現在のデータベース・サーバーの日時を戻す日付関数です。他の列名と同じようにSYSDATEを使用できます。たとえば、表からSYSDATEを選択して、現在の日付を表示できます。通常は、DUALと呼ばれるダミー表からSYSDATEを選択します。

注意: SYSDATEは、データベースが存在するオペレーティング・システムに設定されている現在の日時を戻します。したがって、オーストラリア国内の特定の場所で、米国 (US) 内のリモート・データベースに接続する場合は、sysdate関数によって米国の日時が戻されます。この場合は、セッションのタイムゾーンの現在の日付を戻すCURRENT_DATE関数を使用できます。

CURRENT_DATE関数と他の関連するタイムゾーン関数の詳細は、『Oracle Database 11g: 入門 SQL 基礎 II Ed 1』コースで説明します。

日付の算術演算

- 日付に対して数値の加算または減算を行って日付値を求めます。
- 2つの日付で減算を行って、それらの日付間の日数を求めます。
- 時間数を24で除算し、日付に時間を加算します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付の算術演算

データベースでは日付が数値として格納されるので、加算や減算などの算術演算子を使用して計算を実行できます。日付だけでなく、数値定数の加算と減算も実行できます。

次の演算を実行できます。

演算	結果	説明
日付 + 数値	日付	日数を日付に加算します。
日付 - 数値	日付	日数を日付から減算します。
日付 - 日付	日数	1つの日付を別の日付から減算します。
日付 + 数値/24	日付	時間数を日付に加算します。

日付を使用した算術演算子の使用方法

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

	LAST_NAME	WEEKS
1	King	1041.168239087301587301587301587302
2	Kochhar	923.025381944444444444444444444444
3	De Haan	750.168239087301587301587301587302

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付を使用した算術演算子の使用方法

スライドの例では、部門90のすべての従業員の姓と雇用週数が表示されています。現在の日付 (SYSDATE) から従業員が雇用された日付を引いて結果を7で除算すると、従業員がこれまでに雇用されていた週数を計算できます。

注意: SYSDATEは、現在の日時を戻すSQL関数です。結果は、SQL問合せを実行したときにローカル・データベースのオペレーティング・システムに設定されていた日時によって異なります。

古い日付から新しい日付を減算すると、差は負の数になります。

章の講義項目

- 単一行SQL関数
- 文字関数
- 数値関数
- 日付の操作
- 日付関数

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付操作関数

関数	結果
MONTHS_BETWEEN	2つの日付間の月数
ADD_MONTHS	日付への月の加算
NEXT_DAY	指定された日付の翌日
LAST_DAY	月の最終日
ROUND	日付の丸め
TRUNC	日付の切捨て

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付操作関数

日付関数はOracle日付で動作します。数値を戻すMONTHS_BETWEENを除くすべての日付関数は、DATEデータ型の値を戻します。

- MONTHS_BETWEEN(*date1*, *date2*): *date1*と*date2*間の月数を求めます。結果は正または負の値を取ることができます。*date1*が*date2*よりも後の場合は、結果が正になります。また、*date1*が*date2*よりも前の場合は、結果が負になります。結果の整数以外の部分は、月の一部を表します。
- ADD_MONTHS(*date*, *n*): 月数*n*を*date*に加算します。*n*の値は、整数である必要がありますが、負の数を指定することもできます。
- NEXT_DAY(*date*, '*char*'): 指定した曜日('*char*')で、*date*の後の次の日付を求めます。*char*の値には、日を表す数値または文字列を指定できます。
- LAST_DAY(*date*): *date*が含まれている月の最終日の日付を求めます。

上記のリストは、使用可能な日付関数のサブセットです。ROUNDおよびTRUNC数値関数を使用して、次に示すように日付値を操作することもできます。

- ROUND(*date*[, '*fmt*']): 書式モデル*fmt*で指定した単位に丸めた*date*を戻します。書式モデル*fmt*を省略すると、*date*は最も近い日に丸められます。
- TRUNC(*date*[, '*fmt*']): 日の時刻部分を書式モデル*fmt*で指定された単位に切り捨てた*date*を戻します。書式モデル*fmt*を省略すると、*date*は最も近い日に切り捨てられます。

書式モデルの詳細は、次の章「変換関数と条件式の使用方法」で説明します。

日付関数の使用方法

関数	結果
MONTHS_BETWEEN ('01-SEP-95', '11-JAN-94')	19.6774194
ADD_MONTHS ('31-JAN-96', 1)	'29-FEB-96'
NEXT_DAY ('01-SEP-95', 'FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付関数の使用方法

スライドの例では、ADD_MONTHS関数は、指定された日付値「31-JAN-96」に1か月を加算して「29-FEB-96」を戻しています。この関数は、1996年がうるう年であることを認識して2月の最終日を戻しています。入力日付値を「31-JAN-95」に変更すると、この関数によって「28-FEB-95」が戻されます。

たとえば、雇用月数が100か月未満のすべての従業員に対して、従業員番号、雇用日、雇用月数、6か月審査日、雇用日後の最初の金曜日、雇用月の最終日を表示します。

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
FROM   employees
WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) < 100;
```

	EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY(HIRE_DATE,'FRIDAY')	LAST_DAY(HIRE_DATE)
1	124	16-NOV-99	91.1099600...	16-MAY-00	19-NOV-99	30-NOV-99
2	149	29-JAN-00	88.6906052...	29-JUL-00	04-FEB-00	31-JAN-00
3	178	24-MAY-99	96.8518955...	24-NOV-99	28-MAY-99	31-MAY-99
4	99999	07-JUN-99	96.4002826...	07-DEC-99	11-JUN-99	30-JUN-99
5	113	11-JUN-07	0.25824335...	11-DEC-07	15-JUN-07	30-JUN-07

日付を使用したROUND関数およびTRUNC関数の使用方法

SYSDATE = '25-JUL-03'と仮定します。

関数	結果
ROUND(SYSDATE,'MONTH')	01-AUG-03
ROUND(SYSDATE,'YEAR')	01-JAN-04
TRUNC(SYSDATE,'MONTH')	01-JUL-03
TRUNC(SYSDATE,'YEAR')	01-JAN-03

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付を使用したROUND関数およびTRUNC関数の使用方法

数値と日付値にROUND関数およびTRUNC関数を使用できます。日付と組み合わせて使用すると、これらの関数によって、指定された書式モデルへの丸めまたは切捨てが行われます。したがって、日付を最も近い年または月に丸めることができます。書式モデルが月の場合、1～15の日付は現在の月の最初の日になります。16～31の日付は、翌月の最初の日になります。書式モデルが年の場合、1～6の月は、現在の年の1月1日になります。7～12の月は、翌年の1月1日になります。

例:

1997年に勤務を開始したすべての従業員の雇用日を比較します。ROUND関数およびTRUNC関数を使用して、従業員番号、雇用日、開始月を表示します。

```
SELECT employee_id, hire_date,
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
FROM   employees
WHERE  hire_date LIKE '%97';
```

	EMPLOYEE_ID	HIRE_DATE	ROUND(HIRE_DATE,'MONTH')	TRUNC(HIRE_DATE,'MONTH')
1	142	29-JAN-97	01-FEB-97	01-JAN-97
2	202	17-AUG-97	01-SEP-97	01-AUG-97

まとめ

この章では、次のことを学習しました。

- 関数を使用したデータに対する計算の実行
- 関数を使用したそれぞれのデータ項目の変更

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

単一行関数は任意のレベルにネストできます。単一行関数は、次のデータや値を操作できます。

- 文字データ: LOWER、UPPER、INITCAP、CONCAT、SUBSTR、INSTR、LENGTH
- 数値データ: ROUND、TRUNC、MOD
- 日付値: SYSDATE、MONTHS_BETWEEN、ADD_MONTHS、NEXT_DAY、LAST_DAY

次の点に注意してください。

- 日付値では算術演算子も使用できます。
- 日付値にはROUND関数およびTRUNC関数を使用することもできます。

SYSDATEおよびDUAL

SYSDATEは、現在の日時を戻す日付関数です。通常は、DUALと呼ばれるダミー表からSYSDATEを選択します。

演習3: 概要

この演習では次の項目について説明しています。

- 現在の日付を表示する問合せの記述
- 数値、文字および日付関数を使用する必要がある問合せの作成
- 従業員の勤務年月の計算の実行

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習3: 概要

この演習では、文字、数値および日付データ型に使用できる各種の関数を使用して様々な演習問題に取り組みます。

演習3

パート1

1. システムの日付を表示する問合せを記述します。列にDateというラベルを付けます。

注意: 使用するデータベースがリモートの異なるタイムゾーンにある場合、出力は、データベースが存在するオペレーティング・システムの日付になります。

Date
1 31-MAY-07

2. HR部門は、各従業員の従業員番号、姓、給与、15.5%増額された給与(整数で表示)を表示するレポートを必要としています。列にNew Salaryというラベルを付けます。作成したSQL文をlab_03_02.sqlという名前のファイルに保存します。
3. lab_03_02.sqlファイルで問合せを実行します。

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
1	100	King	24000	27720
2	101	Kochhar	17000	19635
3	102	De Haan	17000	19635
4	103	Hunold	9000	10395
5	104	Ernst	6000	6930
6	107	Lorentz	4200	4851
7	124	Mourgos	5800	6699
8	141	Rajs	3500	4043
9	142	Davies	3100	3581
10	143	Matos	2600	3003

...

19	205	Higgins	12000	13860
20	206	Gietz	8300	9587

4. 問合せlab_03_02.sqlを変更して、新しい給与から前の給与を引いた列を追加します。列にIncreaseというラベルを付けます。ファイルの内容をlab_03_04.sqlとして保存します。修正した問合せを実行します。

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
1	100	King	24000	27720	3720
2	101	Kochhar	17000	19635	2635
3	102	De Haan	17000	19635	2635
4	103	Hunold	9000	10395	1395
5	104	Ernst	6000	6930	930

...

20	206	Gietz	8300	9587	1287
----	-----	-------	------	------	------

演習3(続き)

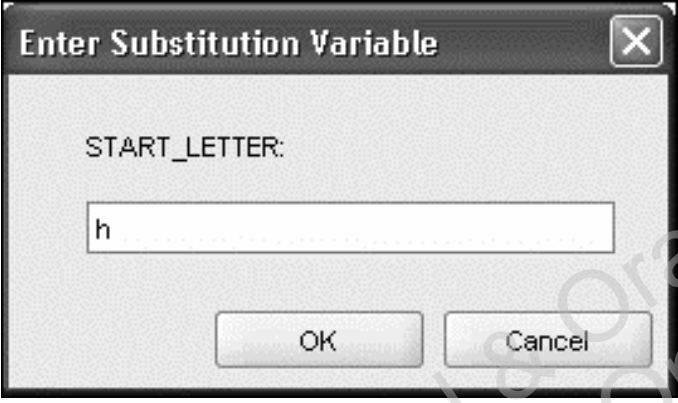
- 名前が文字「J」、「A」または「M」で始まるすべての従業員の姓(最初の文字は大文字、その他のすべての文字は小文字)と姓の長さを表示する問合せを記述します。それぞれの列に適切なラベルを付けます。結果を、従業員の姓でソートします。

	A Z	Name	A Z	Length
1		Abel		4
2		Matos		5
3		Mourgos		7

問合せを記述し直して、姓の先頭に付く文字の入力を求めるプロンプトが表示されるようにします。たとえば、文字の入力を求められたときにユーザーが「H」(大文字)を入力した場合に、姓が「H」で始まるすべての従業員が出力に表示されるようにします。

	A Z	Name	A Z	Length
1		Hartstein		9
2		Higgins		7
3		Hunold		6

問合せを変更して、入力した文字の大文字と小文字の区別が出力に影響しないようにします。入力した文字は、SELECT問合せによって処理される前に、大文字に変換されている必要があります。



	A Z	Name	A Z	Length
1		Hartstein		9
2		Higgins		7
3		Hunold		6

演習3(続き)

6. HR部門は、それぞれの従業員の雇用期間を確認する必要があります。従業員ごとに姓を表示して、その従業員が雇用された日付から今日までの月数を計算します。列に MONTHS_WORKED というラベルを付けます。結果を雇用月数の順に並べます。月数を最も近い整数に丸めます。

注意: この問合せは実行された日付に依存するため、MONTHS_WORKED 列には異なる値が表示されます。

	LAST_NAME	MONTHS_WORKED
1	Zlotkey	88
2	Mourgos	90
3	Grant	96
4	Lorentz	100
5	Vargas	107
6	Taylor	110
7	Matos	111
8	Fay	117
9	Davies	124
10	Abel	133
11	Hartstein	135
12	Rajs	139
13	Higgins	156
14	Gietz	156
15	De Haan	173
16	Ernst	192
17	Hunold	209
18	Kochhar	212
19	Whalen	236
20	King	239

演習3(続き)

時間があるときは、次の演習問題に進みます。

- すべての従業員の姓と給与を表示する問合せを作成します。給与を15文字長に書式設定し、左側に\$記号を埋め込みます。列にSALARYというラベルを付けます。

	LAST_NAME	SALARY
1	King	\$\$\$\$\$\$\$\$\$24000
2	Kochhar	\$\$\$\$\$\$\$\$\$17000

...

20	Gietz	\$\$\$\$\$\$\$\$\$8300
----	-------	------------------------

- 従業員の姓の最初の8文字を表示し、その給与額をアスタリスクで示す問合せを作成します。各アスタリスクは、1,000ドルを表します。データを給与の降順でソートします。列にEMPLOYEES_AND_THEIR_SALARIESというラベルを付けます。

	EMPLOYEES_AND_THEIR_SALARIES
1	King *****
2	Kochhar *****
3	De Haan *****
4	Hartstei *****
5	Higgins *****

...

19	Matos **
20	Vargas **

- 部門90のすべての従業員の姓と雇用週数を表示する問合せを作成します。週数の列にTENUREというラベルを付けます。週数の小数点をゼロに切り捨てます。レコードを、従業員の在職期間の降順で表示します。

注意: TENURE値は、問合せを実行する日付によって異なります。

	LAST_NAME	TENURE
1	King	1041
2	Kochhar	923
3	De Haan	750

4

変換関数と条件式の使用法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

目的

この章を終えると、次のことができるようになります。

- SQLで利用できる各種変換関数の説明
- TO_CHAR、TO_NUMBERおよびTO_DATE変換関数の使用
- SELECT文での条件式の適用

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

この章では、データを1つの型から別の型に変換する関数(文字データから数値データへの変換など)を中心に説明し、SQL SELECT文で使用する条件式について説明します。

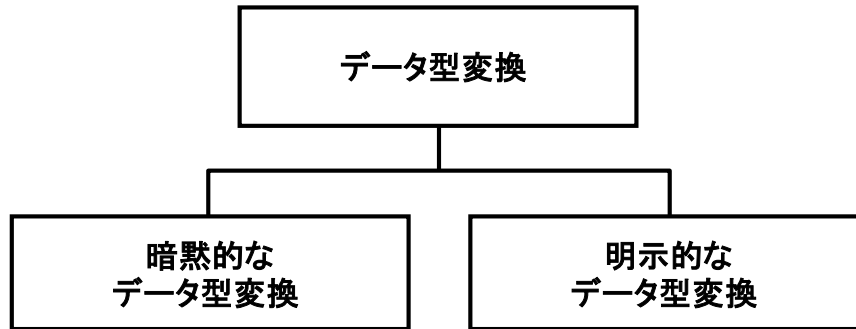
章の講義項目

- 暗黙的および明示的なデータ型変換
- TO_CHAR、TO_DATEおよびTO_NUMBER関数
- 関数のネスト
- 汎用関数:
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件式:
 - CASE
 - DECODE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

変換関数



ORACLE

Copyright © 2007, Oracle. All rights reserved.

変換関数

Oracleデータ型に加え、Oracle Databaseの表の列は、米国規格協会(ANSI)、DB2およびSQL/DSデータ型を使用して定義できます。ただし、Oracleサーバーでは、このようなデータ型は内部的にOracleデータ型に変換されます。

Oracleサーバーは、別のデータ型のデータを想定している場合に、特定のデータ型のデータを受け取る場合があります。この場合、Oracleサーバーで、想定されるデータ型にデータが自動的に変換されます。このデータ型の変換は、Oracleサーバーで暗黙的に実行することも、ユーザーが明示的に実行することもできます。

暗黙的なデータ型変換は、次の2つのスライドで説明する規則に従って動作します。

明示的なデータ型変換は、変換関数を使用して実行します。変換関数は、値を特定のデータ型から別のデータ型に変換します。通常、関数名の形式は *data type* TO *data type* の規則に従います。最初のデータ型は入力データ型で、2番目のデータ型は出力データ型です。

注意: 暗黙的なデータ型変換を使用できますが、明示的なデータ型変換を行ってSQL文の信頼性を確保することをお勧めします。

暗黙的なデータ型変換

Oracleサーバーでは、式によって次の変換を自動的に実行できます。

変換前	変換後
VARCHAR2またはCHAR	NUMBER
VARCHAR2またはCHAR	DATE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

暗黙的なデータ型変換

Oracleサーバーでは、式によってデータ型変換を自動的に実行できます。たとえば、hire_date > '01-JAN-90'という式は、暗黙的に文字列'01-JAN-90'を日付に変換します。したがって、VARCHAR2またはCHAR値は、式によって暗黙的に数値または日付データ型に変換されます。

暗黙的なデータ型変換

式の計算では、Oracleサーバーで自動的に次の変換が実行されます。

変換前	変換後
NUMBER	VARCHAR2またはCHAR
DATE	VARCHAR2またはCHAR

ORACLE

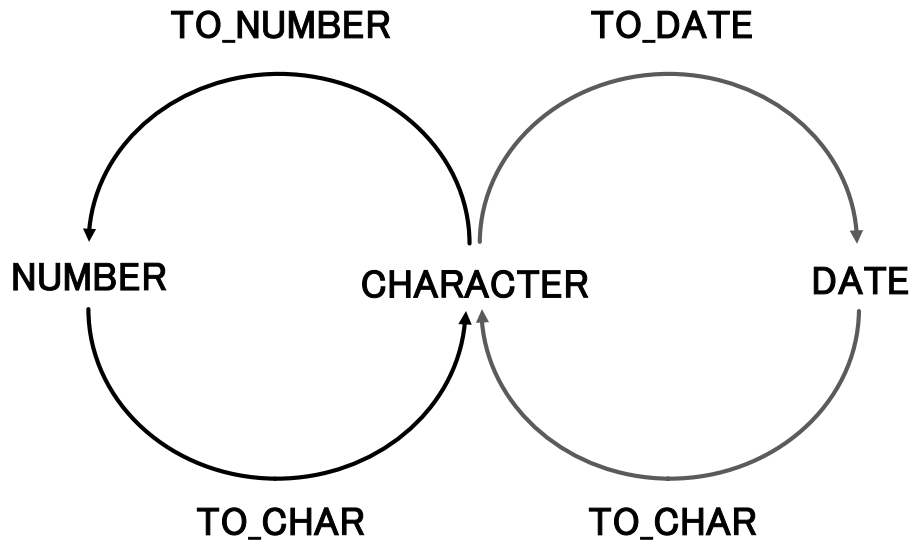
Copyright © 2007, Oracle. All rights reserved.

暗黙的なデータ型変換(続き)

通常、Oracleサーバーではデータ型変換が必要なときに式の規則が使用されます。たとえば、`grade = 2`という式では、数値20000が文字列「2」に暗黙的に変換されます。これは、等級がCHAR(2)列であるためです。

注意: CHARからNUMBERへの変換が成功するのは、文字列が有効な数値を表している場合のみです。

明示的なデータ型変換



ORACLE

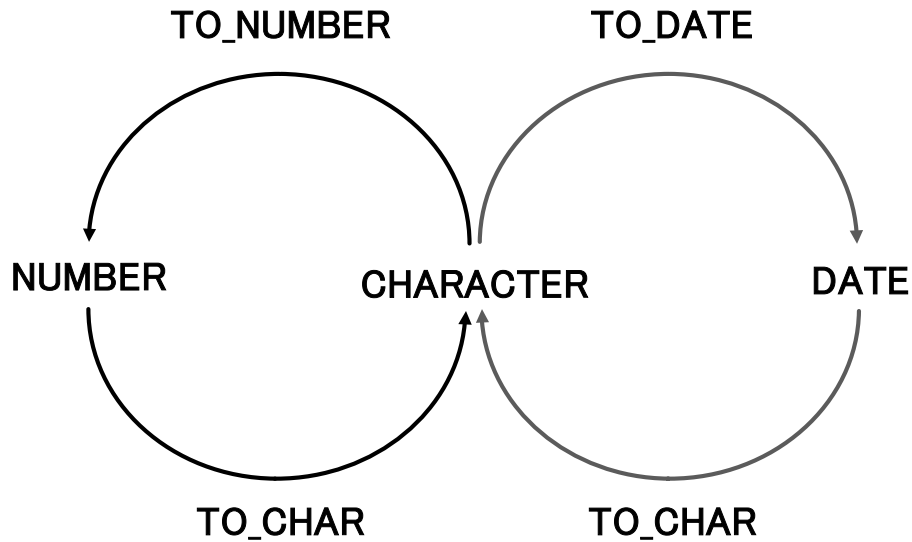
Copyright © 2007, Oracle. All rights reserved.

明示的なデータ型変換

SQLでは、値をあるデータ型から別のデータ型に変換する3つの関数が用意されています。

関数	目的
TO_CHAR(<i>number date</i> ,[<i>fmt</i>],[<i>nlsparms</i>])	<p>書式モデル <i>fmt</i> を使用して、数値または日付値を VARCHAR2 文字列に変換します。</p> <p>数値変換: <i>nlsparms</i> パラメータには、数値書式の要素によって戻される次の文字を指定します。</p> <ul style="list-style-type: none">• 小数点文字• 桁区切り• 各国通貨記号• 国際通貨記号 <p><i>nlsparms</i> などのパラメータが省略されている場合は、この関数によって、セッションのデフォルトのパラメータ値が使用されます。</p>

明示的なデータ型変換



ORACLE

Copyright © 2007, Oracle. All rights reserved.

明示的なデータ型変換(続き)

関数	目的
TO_CHAR(<i>number date</i> , [<i>fmt</i>], [<i>nlsparms</i>])	日付変換: <i>nlsparms</i> パラメータには、月、曜日および省略形が戻されるときに言語を指定します。このパラメータが省略されている場合は、この関数によって、セッションのデフォルトの日付言語が使用されます。
TO_NUMBER(<i>char</i> , [<i>fmt</i>], [<i>nlsparms</i>])	数字が含まれている文字列を、オプションの書式モデル <i>fmt</i> で指定されている書式の数値に変換します。 <i>nlsparms</i> パラメータのこの機能は、数値変換の TO_CHAR関数と同じ目的で使用されます。
TO_DATE(<i>char</i> , [<i>fmt</i>], [<i>nlsparms</i>])	指定された <i>fmt</i> に従って、日付を表す文字列を日付値に変換します。 <i>fmt</i> が省略されている場合、書式は DD-MON-YY になります。 <i>nlsparms</i> パラメータのこの機能は、日付変換の TO_CHAR関数と同じ目的で使用されます。

明示的なデータ型変換(続き)

注意: この章では、使用可能な一部の変換関数についてのみ説明します。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』の変換関数に関する節を参照してください。

Oracle Internal & Oracle Academy
Use Only

章の講義項目

- 暗黙的および明示的なデータ型変換
- TO_CHAR、TO_DATEおよびTO_NUMBER関数
- 関数のネスト
- 汎用関数:
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件式:
 - CASE
 - DECODE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付を使用したTO_CHAR関数の使用方法

```
TO_CHAR(date, 'format_model')  

```

書式モデル:

- 一重引用符で囲む必要があります。
- 大文字と小文字を区別します。
- 任意の有効な日付書式の要素を含めることができます。
- fm要素を使用して、埋め込まれた空白を削除したり先行ゼロを抑制することができます。
- カンマを使用して日付値と区切ります。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付を使用したTO_CHAR関数の使用方法

TO_CHARは、*format_model*で指定された書式で、日時データ型をVARCHAR2データ型の値に変換します。書式モデルは、文字列に格納される日時の書式を説明する文字リテラルです。たとえば、文字列'11-Nov-1999'の日時書式モデルは'DD-Mon-YYYY'です。TO_CHAR関数を使用して、日付をデフォルト書式から指定された書式に変換できます。

ガイドライン

- 書式モデルは一重引用符で囲む必要があり、大文字と小文字が区別されます。
- 書式モデルには、任意の有効な日付書式の要素を含めることができます。ただし、カンマを使用して日付値と書式モデルを区切る必要があります。
- 出力では曜日と月に自動的に空白が埋め込まれます。
- 埋め込まれた空白を削除したり、先行ゼロを抑制したりするには、埋込みモードのfm要素を使用します。

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired  
FROM   employees  
WHERE  last_name = 'Higgins';
```

	EMPLOYEE_ID	MONTH_HIRED
1	205	06/94

日付書式モデルの要素

要素	結果
YYYY	年の完全な数値
YEAR	年のスペル表記(英語)
MM	2桁の月の値
MONTH	月の完全な名前
MON	3文字の月の省略形
DY	3文字の曜日の省略形
DAY	曜日の完全な名前
DD	月の日付

ORACLE

Copyright © 2007, Oracle. All rights reserved.

有効な日付書式の書式要素の例

要素	説明
SCCまたはCC	世紀。サーバーでは、紀元前の日付の先頭に「-」が付けられます。
YYYYまたはSYYYYY (日付に表示される年)	年。サーバーでは、紀元前の日付の先頭に「-」が付けられます。
YYY、YYまたはY	年の下3桁、2桁または1桁。
Y,YYY	この位置にカンマの付いた年。
IYYY、IYY、IY、I	ISO標準に基づく4桁、3桁、2桁または1桁の年。
SYEARまたはYEAR	スペル表記による年。サーバーでは、紀元前の日付の先頭に「-」が付けられます。
BCまたはAD	紀元前または西暦を示します。
B.C.またはA.D.	ピリオドを使用して紀元前または西暦を示します。
Q	年の四半期。
MM	月。2桁の値。
MONTH	空白が埋め込まれた9文字の長さの月の名前。
MON	月の名前。3文字の省略形。
RM	ローマ数字で表した月。
WWまたはW	年または月における週。
DDD、DDまたはD	年、月または週における日。
DAY	空白が埋め込まれた9文字の長さの曜日。
DY	曜日。3文字の省略形。
J	ユリウス日。紀元前4713年12月31日から数えた日数。
IW	ISO標準に基づく、年における週 (1～53)。

日付書式モデルの要素

- 時刻の要素は、日付の時刻の部分を書式設定します。

HH24:MI:SS AM	15:45:32 PM
---------------	-------------

- 文字列は、二重引用符で囲んで追加します。

DD "of" MONTH	12 of OCTOBER
---------------	---------------

- 数字の接尾辞を使用して数字をスペル表記します。

ddspth	fourteenth
--------	------------

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付書式モデルの要素

次の表にリストされている書式を使用すると、時刻情報およびリテラルを表示したり、数値をスペル表記された数値に変換したりすることができます。

要素	説明
AMまたはPM	午前または午後を示す要素。
A.M.またはP.M.	ピリオドを使用した午前または午後を示す要素。
HH、HH12またはHH24	時間、時間 (1～12) または時間 (0～23)
MI	分 (0～59)
SS	秒 (0～59)
SSSSS	午前0時からの経過秒数 (0～86399)

その他の書式

要素	説明
/ . ,	記号は結果に反映されます。
“of the”	引用符で囲まれた文字列は結果に反映されます。

数値表示に影響する接尾辞の指定

要素	説明
TH	序数 (たとえば、4THの場合はDDTH)
SP	スペル表記された数値 (たとえば、FOURの場合はDDSP)
SPTHまたはTHSP	スペル表記された序数 (たとえば、FOURTHの場合はDDSPTH)

Oracle Internal & Oracle Academy
Use Only

日付を使用したTO_CHAR関数の使用方法

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
       AS HIREDATE  
FROM   employees;
```

	LAST_NAME	HIREDATE
1	King	17 June 1987
2	Kochhar	21 September 1989
3	De Haan	13 January 1993
4	Hunold	3 January 1990
5	Ernst	21 May 1991
6	Lorentz	7 February 1999
7	Mourgos	16 November 1999
8	Rajs	17 October 1995
9	Davies	29 January 1997
10	Matos	15 March 1998
...		
19	Higgins	7 June 1994
20	Gietz	7 June 1994

ORACLE

Copyright © 2007, Oracle. All rights reserved.

日付を使用したTO_CHAR関数の使用方法

スライド内のSQL文は、すべての従業員の姓と雇用日を表示します。雇用日は17 June 1987と表示されています。

例:

スライドの例を変更して、日付が「Seventeenth of June 1987 12:00:00 AM」という書式で表示されるようにします。

```
SELECT last_name,  
       TO_CHAR(hire_date,  
               'fmDdsptH "of" Month YYYY fmHH:MI:SS AM')  
       AS HIREDATE  
FROM   employees;
```

	LAST_NAME	HIREDATE
1	King	Seventeenth of June 1987 12:00:00 AM
2	Kochhar	Twenty-First of September 1989 12:00:00 AM

...

指定された書式モデルに従って月が表示されることに注意してください。つまり、最初の文字は大文字、その他の文字は小文字になります。

数値を使用したTO_CHAR関数の使用方法

```
TO_CHAR(number, 'format_model')  
--
```

TO_CHAR関数と組み合わせて、数値を文字として表示できる書式要素をいくつか示します。

要素	結果
9	数値を表します。
0	ゼロを強制表示します。
\$	浮動ドル記号を挿入します。
L	浮動各国通貨記号を使用します。
.	小数点を出力します。
,	カンマを3桁の区切り記号として出力します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

数値を使用したTO_CHAR関数の使用方法

文字列などの数値を操作するときは、NUMBERデータ型の値をVARCHAR2データ型に変換するTO_CHAR関数を使用して、それらの数値を文字データ型に変換する必要があります。このテクニックは特に連結で役立ちます。

数値を使用したTO_CHAR関数の使用方法(続き)

数値の書式要素

数値を文字データ型に変換する場合は、次の書式要素を使用できます。

要素	説明	例	結果
9	数値の位置(9の数で表示幅が決まります)。	999999	1234
0	先行ゼロを表示します。	099999	001234
\$	浮動ドル記号。	\$999999	\$1234
L	浮動各国通貨記号。	L999999	FF1234
DD	指定された位置に小数点を戻します。デフォルトはピリオド(.)です。	99D99	99.99
.	指定された位置に小数点を戻します。	999999.99	1234.00
G	指定された位置に桁区切りを戻します。数値の書式モデルでは、複数の桁区切りを指定できます。	9,999	9G999
,	指定された位置にカンマを戻します。	999,999	1,234
MI	右にマイナス記号が付けられます(負の値)。	999999MI	1234-
PR	負の数をカッコで囲みます。	999999PR	<1234>
EEEE	科学表記法(書式で4つのEの指定が必要)。	99.999EEEE	1.234E+03
U	指定された位置にユーロなどの二重通貨の記号を戻します。	U9999	€1234
V	10の <i>n</i> 倍を乗算します(<i>n</i> はVの後の9の数)。	9999V99	123400
S	負または正の値を戻します。	S9999	-1234または +1234
B	ゼロ値を0ではなく空白として表示します。	B9999.99	1234.00

数値を使用したTO_CHAR関数の使用方法

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM   employees  
WHERE  last_name = 'Ernst';
```

	SALARY
1	\$6,000.00

ORACLE

Copyright © 2007, Oracle. All rights reserved.

数値を使用したTO_CHAR関数の使用方法(続き)

- Oracleサーバーでは、書式モデルで指定された桁数を超える整数は、数値記号(#)の文字列で表示されます。
- Oracleサーバーでは、格納されている10進値は、書式モデルで指定された小数位に丸められます。

TO_NUMBERおよびTO_DATE関数の使用方法

- TO_NUMBER関数を使用して、文字列を数値書式に変換します。

```
TO_NUMBER(char[, 'format_model'])
```

- TO_DATE関数を使用して、文字列を日付書式に変換します。

```
TO_DATE(char[, 'format_model'])
```

- これらの関数では、fx修飾子を使用できます。この修飾子は、TO_DATE関数の文字引数および日付書式モデルの完全一致を指定します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

TO_NUMBERおよびTO_DATE関数の使用方法

文字列を数値または日付に変換することができます。このタスクを実行するには、TO_NUMBERまたはTO_DATE関数を使用します。書式モデルは、前に実行した書式要素に基づいて選択します。

fx修飾子は、TO_DATE関数の文字引数および日付書式モデルの完全一致を指定します。

- 文字引数内の記号および引用符付きテキストは、書式モデルの対応する部分と完全に一致する必要があります(大文字と小文字の区別を除く)。
- 文字引数に余分な空白を含めることはできません。fxを使用しない場合、Oracleサーバーは余分な空白を無視します。
- 文字引数内の数値データは、書式モデル内の対応する要素と同じ桁数になっている必要があります。fxを使用しない場合、文字引数内の数では先行ゼロを省略できます。

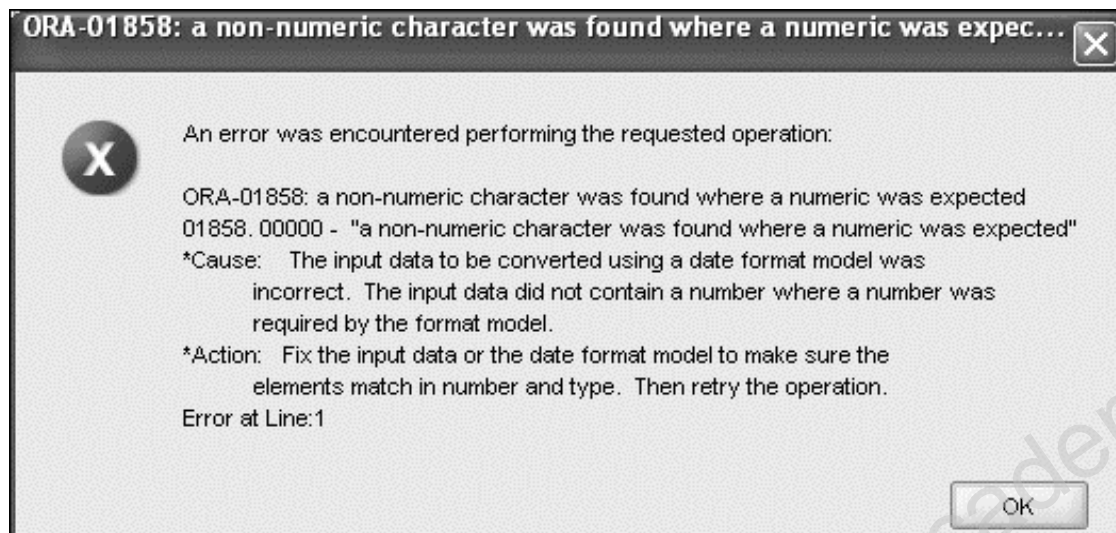
TO_NUMBERおよびTO_DATE関数の使用方法(続き)

例:

1999年5月24日に勤務を開始したすべての従業員の名前と雇用日を表示します。次の例では、月 *May* の後に2つの空白があり、その後に数値24が続いています。fx修飾子が使用されているため、完全一致が要求され、*May* の後ろの空白は認識されません。

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May   24, 1999', 'fxMonth DD, YYYY');
```

エラー:



RR日付書式を使用した TO_CHARおよびTO_DATE関数の使用方法

1990年より前に雇用された従業員を検索するには、RR日付書式を使用します。この書式では、コマンドが1999年に実行された場合でも現在実行された場合でも同じ結果が戻されます。

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM   employees
WHERE  hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	King	17-Jun-1987
2	Kochhar	21-Sep-1989
3	Whalen	17-Sep-1987

ORACLE

Copyright © 2007, Oracle. All rights reserved.

RR日付書式を使用したTO_CHARおよびTO_DATE関数の使用方法

1990年よりも前に雇用された従業員を検索するには、RR書式を使用できます。現在の年は1999よりも大きいので、RR書式は日付の年部分を1950～1999の範囲で解析します。

一方、次のコマンドでは、行が選択されません。YY書式は、現在の世紀(2090)で日付の年部分を解析するためです。

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM   employees
WHERE  TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

```
0 rows selected
```


章の講義項目

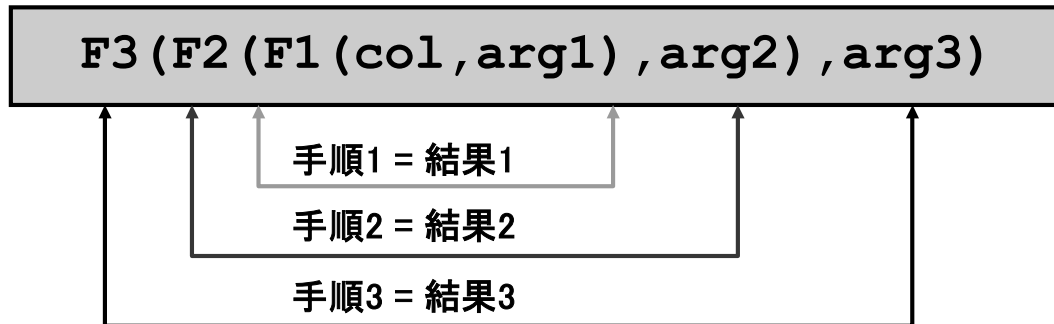
- 暗黙的および明示的なデータ型変換
- TO_CHAR、TO_DATEおよびTO_NUMBER関数
- 関数のネスト
- 汎用関数:
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件式:
 - CASE
 - DECODE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

関数のネスト

- 単一行関数は任意のレベルにネストできます。
- ネストされた関数は、最も深いレベルから最も浅いレベルへと順番に計算されます。



ORACLE

Copyright © 2007, Oracle. All rights reserved.

関数のネスト

単一行関数は任意の深さにネストできます。ネストされた関数は、最も内側のレベルから最も外側のレベルへと順番に計算されます。以降の例に、これらの関数の柔軟性を示します。

関数のネスト

```
SELECT last_name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))  
FROM   employees  
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

ORACLE

Copyright © 2007, Oracle. All rights reserved.

関数のネスト(続き)

スライドの例では、部門60の従業員の姓が表示されています。SQL文の計算では、次の3つの手順が実行されます。

1. 内側の関数によって、姓の最初の8文字が取得されます。

```
Result1 = SUBSTR (LAST_NAME, 1, 8)
```

2. その外側にある関数によって、結果と_USが連結されます。

```
Result2 = CONCAT (Result1, '_US')
```

3. 最も外側の関数によって、結果が大文字に変換されます。

列別名が指定されていないため、式全体が列ヘッダーになります。

例:

雇用日から6か月後の次の金曜日の日付を表示します。結果には、日付がFriday, August 13th, 1999と表示される必要があります。結果を雇用日の順序で並べます。

```
SELECT      TO_CHAR(NEXT_DAY(ADD_MONTHS  
                           (hire_date, 6), 'FRIDAY'),  
              'fmDay, Month ddth, YYYY')  
            "Next 6 Month Review"  
FROM        employees  
ORDER BY    hire_date;
```

章の講義項目

- 暗黙的および明示的なデータ型変換
- TO_CHAR、TO_DATEおよびTO_NUMBER関数
- 関数のネスト
- 汎用関数:
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件式:
 - CASE
 - DECODE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

汎用関数

次の関数は任意のデータ型で動作し、NULLの使用に関係します。

- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

ORACLE

Copyright © 2007, Oracle. All rights reserved.

汎用関数

これらの関数は任意のデータ型で動作し、式リストのNULL値の使用に関係します。

関数	説明
NVL	NULL値を実値に変換します。
NVL2	expr1がNULLでない場合、NVL2はexpr2を返します。expr1がNULLの場合、NVL2はexpr3を返します。引数expr1には任意のデータ型を使用できます。
NULLIF	2つの式を比較して、それらの式が等しい場合はNULLを返します。異なる場合は、最初の式を返します。
COALESCE	式リスト内のNULLではない最初の式を返します。

注意: 使用可能な各種関数の詳細は、『Oracle Database SQL言語リファレンス11gリリース1 (11.1)』の関数に関する節を参照してください。

NVL関数

NULL値を実値に変換します。

- 使用できるデータ型は、日付、文字および数値です。
- データ型は、次のいずれかである必要があります。
 - NVL(commission_pct,0)
 - NVL(hire_date,'01-JAN-97')
 - NVL(job_id,'No Job Yet')

ORACLE

Copyright © 2007, Oracle. All rights reserved.

NVL関数

NULL値を実値に変換するには、NVL関数を使用します。

構文

NVL (*expr1*, *expr2*)

構文の内容:

- *expr1*は、NULLが含まれている可能性のあるソースの値または式です。
- *expr2*は、NULLを変換するためのターゲット値です。

NVL関数を使用して任意のデータ型を変換できますが、戻り値は常に*expr1*のデータ型と同じになります。

様々なデータ型のNVL変換

データ型	変換例
NUMBER	NVL(<i>number_column</i> ,9)
DATE	NVL(<i>date_column</i> , '01-JAN-95')
CHARまたはVARCHAR2	NVL(<i>character_column</i> , 'Unavailable')

NVL関数の使用方法

```
SELECT last name, salary, NVL(commission_pct, 0),
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200
11	Vargas	2500	0	30000
12	Zlotkey	10500	0.2	151200

...

1 2

ORACLE

Copyright © 2007, Oracle. All rights reserved.

NVL関数の使用方法

すべての従業員の年間所得を計算するには、月給に12を乗算して、その結果に歩合の割合を加算する必要があります。

```
SELECT last_name, salary, commission_pct,
       (salary*12) + (salary*12*commission_pct) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
1	King	24000	(null)	(null)

...

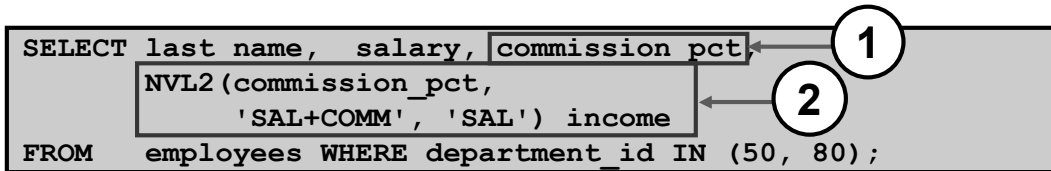
11	Vargas	2500	(null)	(null)
12	Zlotkey	10500	0.2	151200
13	Abel	11000	0.3	171600

...

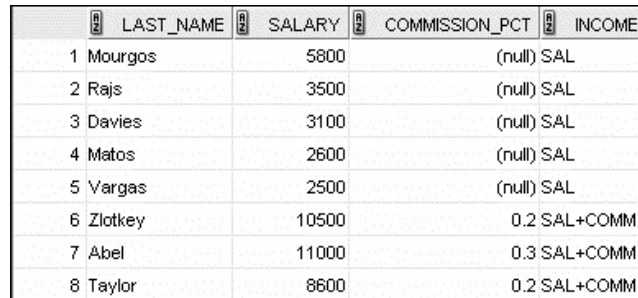
年間所得は歩合を受け取る従業員に対してのみ計算されることに注意してください。式内の列の値がNULLの場合、結果はNULLになります。すべての従業員の値を計算するには、NULL値を数値に変換してから、算術演算子を適用する必要があります。スライドの例では、NVL関数を使用してNULL値をゼロに変換しています。

NVL2関数の使用方法

```
SELECT last name, salary, commission_pct  
      NVL2(commission_pct,  
            'SAL+COMM', 'SAL') income  
FROM   employees WHERE department_id IN (50, 80);
```



	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM



ORACLE

Copyright © 2007, Oracle. All rights reserved.

NVL2関数の使用方法

NVL2関数は、最初の式を確認します。最初の式がNULLでない場合は、NVL2関数は2番目の式を返します。最初の式がNULLの場合は、3番目の式が返されます。

構文

`NVL2(expr1, expr2, expr3)`

構文の内容:

- *expr1*は、NULLが含まれている可能性のあるソースの値または式です。
- *expr2*は、*expr1*がNULLでない場合に返される値です。
- *expr3*は、*expr1*がNULLの場合に返される値です。

スライドの例では、COMMISSION_PCT列が確認されます。値が検出されると、2番目の式のSAL+COMMが返されます。COMMISSION_PCT列にNULL値が存在する場合は、3番目の式のSALが返されます。

引数*expr1*には任意のデータ型を使用できます。引数*expr2*および*expr3*には、LONG以外の任意のデータ型を使用できます。*expr2*と*expr3*のデータ型が異なる場合は、Oracleサーバーによって、*expr3*が*expr2*のデータ型に変換されてから比較が行われます。ただし、*expr3*がNULL定数の場合を除きます。NULL定数の場合、データ型の変換は必要ありません。戻り値のデータ型は常に*expr2*のデータ型と同じになります。ただし、*expr2*が文字データの場合は、戻り値のデータ型はVARCHAR2になります。

NULLIF関数の使用方法

SQL Query:

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name, LENGTH(last_name) "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM employees;
```

Result Table:

	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9
...					
19	Jennifer	8	Whalen	6	8
20	Eleni	5	Zlotkey	7	5

Copyright © 2007, Oracle. All rights reserved.

NULLIF関数の使用方法

NULLIF関数は2つの式を比較します。2つの式が等しい場合、この関数はNULLを戻します。式が異なる場合は、最初の式を戻します。ただし、最初の式にはリテラルNULLを指定できません。

構文

```
NULLIF (expr1, expr2)
```

構文の内容:

- NULLIFは`expr1`と`expr2`を比較します。2つの式が等しい場合、この関数はNULLを戻します。式が異なる場合は、`expr1`を戻します。ただし、`expr1`にはリテラルNULLを指定できません。

スライドの例では、EMPLOYEES表の名前の長さ、EMPLOYEES表の姓の長さが比較されています。これらの名前の長さが等しい場合は、NULL値が表示されます。長さが異なる場合は、名前の長さが表示されます。

注意: NULLIF関数は、論理的には次のCASE式と同等です。CASE式については、この後のページで説明します。

```
CASE WHEN expr1 = expr 2 THEN NULL ELSE expr1 END
```

COALESCE関数の使用方法

- COALESCE関数は、複数の代替値を取れるという点で NVL関数よりも便利です。
- 最初の式がNULLでない場合、COALESCE関数はその式を返します。NULLの場合は、他のすべての式に対して COALESCEを実行します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

COALESCE関数の使用方法

COALESCE関数は、リスト内のNULLでない最初の式を返します。

構文

```
COALESCE (expr1, expr2, ... exprn)
```

構文の内容:

- *expr1*は、この式がNULLでない場合にこの式を返します。
- *expr2*は、最初の式がNULLで、この式がNULLでない場合にこの式を返します。
- *exprn*は、先行する式がNULLの場合にこの式を返します。

すべての式において、データ型は同じである必要があることに注意してください。

COALESCE関数の使用方法

```
SELECT last name, employee id,  
COALESCE(TO_CHAR(commission_pct), TO_CHAR(manager_id),  
          'No commission and no manager')  
FROM employees;
```

	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COM
1	King	100	No commission and no manager
2	Kochhar	101	100
3	De Haan	102	100
4	Hunold	103	102
5	Ernst	104	103
6	Lorentz	107	103
7	Mourgos	124	100
8	Rajs	141	124

...

12	Zlotkey	149	.2
13	Abel	174	.3
14	Taylor	176	.2
15	Grant	178	.15
16	Whalen	200	101

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

COALESCE関数の使用方法(続き)

スライドの例では、manager_id値がNULLでない場合は、値が表示されます。manager_id値がNULLの場合は、commission_pctが表示されます。manager_idおよびcommission_pct値がNULLの場合は、「No commission and no manager」と表示されます。TO_CHAR関数が適用されて、すべての式が同じデータ型になっていることに注意してください。

COALESCE関数の使用方法(続き)

例:

歩合を受け取らない従業員に対して、組織は給与を\$2,000増額する必要があります。また、歩合を受け取る従業員に対しては、既存の給与と歩合の額を加算した新しい給与を計算する問合せを必要としています。

```
SELECT last_name, salary, commission_pct,  
       COALESCE((salary+(commission_pct*salary)), salary+2000, salary)  
       "New Salary"  
FROM   employees;
```

注意: 出力を確認してください。歩合を受け取らない従業員の場合は、New Salary列に\$2,000増額された給与が表示され、歩合を受け取る従業員の場合は、New Salary列に給与に歩合の額を加算した額が表示されます。

	A Z	LAST_NAME	A Z	SALARY	A Z	COMMISSION_PCT	A Z	New Salary
1		King		24000		(null)		26000
2		Kochhar		17000		(null)		19000
3		De Haan		17000		(null)		19000
4		Hunold		9000		(null)		11000

...

9		Davies		3100		(null)		5100
10		Matos		2600		(null)		4600
11		Vargas		2500		(null)		4500
12		Zlotkey		10500		0.2		12600
13		Abel		11000		0.3		14300
14		Taylor		8600		0.2		10320
15		Grant		7000		0.15		8050
16		Whalen		4400		(null)		6400
17		Hartstein		13000		(null)		15000
18		Fay		6000		(null)		8000
19		Higgins		12000		(null)		14000
20		Gietz		8300		(null)		10300

...

章の講義項目

- 暗黙的および明示的なデータ型変換
- TO_CHAR、TO_DATEおよびTO_NUMBER関数
- 関数のネスト
- 汎用関数:
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件式:
 - CASE
 - DECODE

ORACLE

Copyright © 2007, Oracle. All rights reserved.

条件式

- SQL文でIF-THEN-ELSE論理を使用できるようにします。
- 次の2つのメソッドを使用します。
 - CASE式
 - DECODE関数

ORACLE

Copyright © 2007, Oracle. All rights reserved.

条件式

SQL文に条件処理 (IF-THEN-ELSE論理) を実装するために、CASE式とDECODE関数の2つのメソッドが使用されます。

注意: CASE式はANSI SQLに準拠しています。DECODE関数はOracle構文に固有のものです。

CASE式

IF-THEN-ELSE文の機能を実行すると、条件付きの問合せを簡単に実行できます。

```
CASE expr WHEN comparison_expr1 THEN return_expr1  
      [WHEN comparison_expr2 THEN return_expr2  
      WHEN comparison_exprn THEN return_exprn  
      ELSE else_expr]  
END
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

CASE式

CASE式では、プロシージャを呼び出さずに、SQL文でIF-THEN-ELSE論理を使用できます。

単純なCASE式では、Oracleサーバーによって、*expr*が*comparison_expr*と等しい最初のWHEN ... THENの組合せが検索されて、*return_expr*が戻されます。この条件に一致するWHEN ... THENの組合せがない場合、およびELSE句が存在する場合は、Oracleサーバーによって*else_expr*が戻されます。それ以外の場合は、OracleサーバーによってNULLが戻されます。*return_expr*および*else_expr*にリテラルNULLを指定することはできません。

すべての式(*expr*、*comparison_expr*および*return_expr*)は、同じデータ型である必要があります。データ型には、CHAR、VARCHAR2、NCHARまたはNVARCHAR2のいずれかを使用できます。

CASE式の使用法

IF-THEN-ELSE文の機能を実行すると、条件付きの問合せを簡単に実行できます。

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                  WHEN 'ST_CLERK' THEN 1.15*salary  
                  WHEN 'SA_REP' THEN 1.20*salary  
                  ELSE salary END "REVISED_SALARY"  
FROM employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

ORACLE

Copyright © 2007, Oracle. All rights reserved.

CASE式の使用法

スライドのSQL文では、JOB_IDの値がデコードされます。JOB_IDがIT_PROGの場合、給与の増額は10%です。JOB_IDがST_CLERKの場合、給与の増額は15%です。また、JOB_IDがSA_REPの場合、給与の増額は20%です。その他のすべての職務では、給与の増額はありません。

DECODE関数を使用して、同じ文を記述することができます。

次に、検索CASE式の例を示します。検索CASE式では、リストされた条件が検出されるまで左から右へ検索が実行され、その後に戻り式が戻されます。真となる条件がない場合、およびELSE句が存在する場合は、ELSE句の戻り式が戻されます。それ以外の場合は、NULLが戻されます。

```
SELECT last_name, salary,  
       (CASE WHEN salary<5000 THEN 'Low'  
            WHEN salary<10000 THEN 'Medium'  
            WHEN salary<20000 THEN 'Good'  
            ELSE 'Excellent'  
       END) qualified_salary  
FROM employees;
```


DECODE関数

CASE式またはIF-THEN-ELSE文の機能を実行すると、条件付きの問合せを簡単に実行できます。

```
DECODE(col|expression, search1, result1  
      [, search2, result2, ..., ]  
      [, default])
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

DECODE関数

DECODE関数は、様々な言語で使用されるIF-THEN-ELSE論理と同じような方法で式をデコードします。DECODE関数は、*expression*をデコードする前に、各 *search* 値と比較します。式が *search* と同じ場合は、*result* が戻されます。

デフォルト値が省略されている場合は、NULL値が戻され、検索値はいずれの結果値にも一致しません。

DECODE関数の使用方法

```
SELECT last name, job id, salary,  
       DECODE(job_id, 'IT_PROG', 1.10*salary,  
                 'ST_CLERK', 1.15*salary,  
                 'SA_REP', 1.20*salary,  
                 salary)  
       REVISED_SALARY  
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

ORACLE

Copyright © 2007, Oracle. All rights reserved.

DECODE関数の使用方法

スライドのSQL文では、JOB_IDの値がテストされます。JOB_IDがIT_PROGの場合、給与の増額は10%です。JOB_IDがST_CLERKの場合、給与の増額は15%です。また、JOB_IDがSA_REPの場合、給与の増額は20%です。その他のすべての職務では、給与の増額はありません。

次のように、同じ文を、擬似コードでIF-THEN-ELSEとして表すことができます。

```
IF job_id = 'IT_PROG' THEN salary = salary*1.10  
IF job_id = 'ST_CLERK' THEN salary = salary*1.15  
IF job_id = 'SA_REP' THEN salary = salary*1.20  
ELSE salary = salary
```

DECODE関数の使用方法

部門80の各従業員の適用可能な税率を表示します。

```
SELECT last name, salary,  
       DECODE (TRUNC(salary/2000, 0),  
               0, 0.00,  
               1, 0.09,  
               2, 0.20,  
               3, 0.30,  
               4, 0.40,  
               5, 0.42,  
               6, 0.44,  
               0.45) TAX_RATE  
FROM   employees  
WHERE  department_id = 80;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

DECODE関数の使用方法(続き)

このスライドはDECODE関数を使用した別の例を示しています。この例では、月給に基づいて、部門80の各従業員の税率を特定します。税率は次のとおりです。

月給の範囲	税率
\$0.00–1,999.99	00%
\$2,000.00–3,999.99	09%
\$4,000.00–5,999.99	20%
\$6,000.00–7,999.99	30%
\$8,000.00–9,999.99	40%
\$10,000.00–11,999.99	42%
\$12,200.00–13,999.99	44%
\$14,000.00以上	45%

	LAST_NAME	SALARY	TAX_RATE
1	Zlotkey	10500	0.42
2	Abel	11000	0.42
3	Taylor	8600	0.4

まとめ

この章では、次のことを学習しました。

- 関数を使用した表示用の日付書式の変更
- 関数を使用した列のデータ型の変換
- NVL関数の使用
- SELECT文でのIF-THEN-ELSE論理とその他の条件式の使用

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

次の点に注意してください。

- 変換関数TO_CHAR、TO_DATE、TO_NUMBERは、文字、日付および数値の変換を実行できます。
- NVL、NVL2、NULLIF、COALESCEなど、NULLの使用に関係する複数の関数があります。
- CASE式またはDECODE関数を使用して、SQL文にIF-THEN-ELSE論理を適用できます。

演習4: 概要

この演習では次の項目について説明しています。

- TO_CHAR、TO_DATEなどのDATE関数を使用する問合せの作成
- DECODEやCASEなどの条件式を使用する問合せの作成

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習4: 概要

この演習では、TO_CHAR関数、TO_DATE関数、およびDECODEやCASEなどの条件式を使用して様々な演習問題に取り組みます。ネストされた関数では、結果が内側から外側の関数へと順番に計算されることに注意してください。

演習4

- 各従業員の次のデータを生成するレポートを作成します。
 <employee last name> earns <salary> monthly but wants <3 times salary.>。列にDream Salariesというラベルを付けます。

	A2	Dream Salaries
1	King	earns \$24,000.00 monthly but wants \$72,000.00.
2	Kochhar	earns \$17,000.00 monthly but wants \$51,000.00.
3	De Haan	earns \$17,000.00 monthly but wants \$51,000.00.
4	Hunold	earns \$9,000.00 monthly but wants \$27,000.00.
5	Ernst	earns \$6,000.00 monthly but wants \$18,000.00.

...

19	Higgins	earns \$12,000.00 monthly but wants \$36,000.00.
20	Gietz	earns \$8,300.00 monthly but wants \$24,900.00.

- 各従業員の姓、雇用日、および6か月の雇用期間後の最初の月曜日となる給与審査日を表示します。列にREVIEWというラベルを付けます。「Monday, the Thirty-First of July, 2000」のような書式で表示されるように日付の書式を設定します。

	A2	LAST_NAME	HIRE_DATE	A2	REVIEW
1	King		17-JUN-87		Monday, the Twenty-First of December, 1987
2	Kochhar		21-SEP-89		Monday, the Twenty-Sixth of March, 1990
3	De Haan		13-JAN-93		Monday, the Nineteenth of July, 1993
4	Hunold		03-JAN-90		Monday, the Ninth of July, 1990
5	Ernst		21-MAY-91		Monday, the Twenty-Fifth of November, 1991

...

19	Higgins		07-JUN-94		Monday, the Twelfth of December, 1994
20	Gietz		07-JUN-94		Monday, the Twelfth of December, 1994

演習4(続き)



3. 従業員の姓、雇用日および従業員が勤務を開始した曜日を表示します。列にDAYというラベルを付けます。月曜日から開始されるように、曜日で結果をソートします。

	 LAST_NAME	HIRE_DATE	 DAY
1	Grant	24-MAY-99	MONDAY
2	Gietz	07-JUN-94	TUESDAY
3	Taylor	24-MAR-98	TUESDAY
4	Higgins	07-JUN-94	TUESDAY
5	Rajs	17-OCT-95	TUESDAY

...

19	Lorentz	07-FEB-99	SUNDAY
20	Fay	17-AUG-97	SUNDAY

4. 従業員の姓と歩合の額を表示する問合せを作成します。従業員が歩合を受け取らない場合は、「No Commission」を表示します。列にCOMMというラベルを付けます。

	 LAST_NAME	 COMM
1	King	No Commission
2	Kochhar	No Commission
3	De Haan	No Commission
4	Hunold	No Commission
5	Ernst	No Commission
6	Lorentz	No Commission

...

12	Zlotkey	.2
13	Abel	.3
14	Taylor	.2
15	Grant	.15
16	Whalen	No Commission
17	Hartstein	No Commission
18	Fay	No Commission
19	Higgins	No Commission
20	Gietz	No Commission

演習4(続き)

時間があるときは、次の演習問題に進みます。

5. DECODE関数で次のデータを使用して、列JOB_IDの値に基づくすべての従業員の等級を表示する問合せを作成します。

職務	等級
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
上記のいずれでもない	0

	R2	JOB_ID	R2	GRADE
1		AC_ACCOUNT	0	
2		AC_MGR	0	
3		AD_ASST	0	
4		AD_PRES	A	
5		AD_VP	0	

...

18		ST_CLERK	E
19		ST_CLERK	E
20		ST_MAN	B

6. CASE構文を使用して、前の演習問題の文を記述し直します。

	R2	JOB_ID	R2	GRADE
1		AC_ACCOUNT	0	
2		AC_MGR	0	
3		AD_ASST	0	
4		AD_PRES	A	
5		AD_VP	0	

...

18		ST_CLERK	E
19		ST_CLERK	E
20		ST_MAN	B

5

グループ関数を使用した 集計データのレポート

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

目的

この章を終えると、次のことができるようになります。

- 使用可能なグループ関数の理解
- グループ関数の使用方法の説明
- GROUP BY句を使用したデータのグループ化
- HAVING句を使用したグループ化された行の組込みまたは除外

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

この章では関数についてさらに学習します。行グループのサマリー情報(平均など)の取得を中心に説明します。表内の行を小さいセットにグループ化する方法と、行のグループの検索基準を指定する方法について説明します。

章の講義項目

- グループ関数:
 - タイプと構文
 - AVG、SUM、MIN、MAX、COUNTの使用
 - グループ関数内でのDISTINCTキーワードの使用
 - グループ関数内のNULL値
- 行のグループ化:
 - GROUP BY句
 - HAVING句
- グループ関数のネスト

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数の概要

グループ関数は行のセットに対して動作し、グループごとに1つの結果を返します。

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	90	24000
2	90	17000
3	90	17000
4	60	9000
5	60	6000
6	60	4200
7	50	5800
8	50	3500
9	50	3100
10	50	2600
...		
18	20	6000
19	110	12000
20	110	8300

EMPLOYEES表の
最大給与

MAX(SALARY)
24000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数の概要

単一行関数と異なり、グループ関数は行のセットに対して動作し、グループごとに1つの結果を返します。これらのセットによって、表全体またはグループに分かれた表が構成されます。

グループ関数のタイプ

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数のタイプ

各関数は、引数を受け入れます。次の表に、構文で使えるオプションを示します。

関数	説明
AVG([DISTINCT <u>ALL</u>] <i>n</i>)	<i>n</i> の平均値。NULL値は無視されます。
COUNT(({* [DISTINCT <u>ALL</u>] <i>expr</i> })	行数。 <i>expr</i> を使用すると、NULL以外の値が対象となります(*を使用すると、重複とNULLを持つ行を含め、選択されたすべての行がカウントされます)。
MAX([DISTINCT <u>ALL</u>] <i>expr</i>)	<i>expr</i> の最大値。NULL値は無視されます。
MIN([DISTINCT <u>ALL</u>] <i>expr</i>)	<i>expr</i> の最小値。NULL値は無視されます。
STDDEV([DISTINCT <u>ALL</u>] <i>x</i>)	<i>n</i> の標準偏差。NULL値は無視されます。
SUM([DISTINCT <u>ALL</u>] <i>n</i>)	<i>n</i> の合計値。NULL値は無視されます。
VARIANCE([DISTINCT <u>ALL</u>] <i>x</i>)	<i>n</i> の分散。NULL値は無視されます。

グループ関数: 構文

```
SELECT    group_function(column), ...
FROM      table
[WHERE    condition]
[ORDER BY column];
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数: 構文

グループ関数はSELECTキーワードの後に指定します。複数のグループ関数をカンマで区切って指定できます。

グループ関数を使用するためのガイドライン:

- DISTINCTを指定すると、関数によって重複のない値のみが考慮されます。ALLを指定すると、重複を含むすべての値が考慮されます。デフォルトでALLが設定されているため、指定する必要はありません。
- *expr*引数を持つ関数のデータ型として、CHAR、VARCHAR2、NUMBERまたはDATEを使用できます。
- すべてのグループ関数においてNULL値は無視されます。NULL値の値を置換するには、NVL、NVL2またはCOALESCE関数を使用します。

AVGおよびSUM関数の使用方法

AVGとSUMは数値データに使用できます。

```
SELECT AVG(salary), MAX(salary),  
       MIN(salary), SUM(salary)  
FROM   employees  
WHERE  job_id LIKE '%REP%';
```

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

ORACLE

Copyright © 2007, Oracle. All rights reserved.

AVGおよびSUM関数の使用方法

AVG、SUM、MINおよびMAX関数は、数値データを格納できる列に対して使用できます。スライドの例では、すべての販売担当者の月給の平均、最高額、最低額および合計が表示されています。

MINおよびMAX関数の使用方法

MINとMAXは、数値、文字および日付データ型に使用できます。

```
SELECT MIN(hire_date), MAX(hire_date)
FROM   employees;
```

MIN(HIRE_DATE)	MAX(HIRE_DATE)
1 17-JUN-87	29-JAN-00

ORACLE

Copyright © 2007, Oracle. All rights reserved.

MINおよびMAX関数の使用方法

MAXおよびMIN関数は、数値、文字および日付データ型に使用できます。スライドの例では、最も雇用期間の短い従業員と最も雇用期間の長い従業員が表示されています。

次の例では、すべての従業員のアルファベット順のリスト内に表示されている、最初の従業員と最後の従業員の姓が表示されます。

```
SELECT MIN(last_name), MAX(last_name)
FROM   employees;
```

	MIN(LAST_NAME)	MAX(LAST_NAME)
1	Abel	Zlotkey

注意: AVG、SUM、VARIANCEおよびSTDDEV関数では、数値データ型のみを使用できます。MAXとMINでは、LOBまたはLONGデータ型は使用できません。

COUNT関数の使用方法

COUNT(*)は、表内の行数を戻します。

①

```
SELECT COUNT (*)  
FROM employees  
WHERE department_id = 50;
```

COUNT(*)
5

COUNT(expr)は、*expr*のNULLではない値を持つ行の数を戻します。

②

```
SELECT COUNT (commission_pct)  
FROM employees  
WHERE department_id = 80;
```

COUNT(COMMISSION_PCT)
3

ORACLE

Copyright © 2007, Oracle. All rights reserved.

COUNT関数の使用方法

COUNT関数には次の3つの書式があります。

- COUNT(*)
- COUNT(*expr*)
- COUNT(DISTINCT *expr*)

COUNT(*)は、重複行や、列のいずれかにNULL値が含まれる行など、SELECT文の基準を満たす表内の行数を戻します。SELECT文にWHERE句が含まれている場合、COUNT(*)は、WHERE句の条件を満たす行の数を戻します。

また、COUNT(*expr*)は、*expr*によって識別される列のNULLではない値の数を戻します。

COUNT(DISTINCT *expr*)は、*expr*によって識別される列のNULLではない一意の値の数を戻します。

例:

1. スライドの例では、部門50の従業員の数が表示されています。
2. スライドの例では、歩合を受け取ることができる部門80の従業員の数が表示されています。

DISTINCTキーワードの使用方法

- COUNT(DISTINCT expr)は、*expr*のNULLではない一意の値の数を返します。
- EMPLOYEES表の一意の部門値の数を表示するには、次のように指定します。

```
SELECT COUNT(DISTINCT department_id)
FROM employees;
```

COUNT(DISTINCTDEPARTMENT_ID)	
1	7

ORACLE

Copyright © 2007, Oracle. All rights reserved.

DISTINCTキーワードの使用方法

DISTINCTキーワードを使用すると、列内の重複値のカウントを抑制できます。
スライドの例では、EMPLOYEES表内の一意の部門値の数が表示されています。

グループ関数とNULL値

グループ関数は、列内のNULL値を無視します。

①

```
SELECT AVG(commission_pct)
FROM employees;
```

AVG(COMMISSION_PCT)	
1	0.2125

NVL関数は、グループ関数でNULL値が含まれるようにします。

②

```
SELECT AVG(NVL(commission_pct, 0))
FROM employees;
```

AVG(NVL(COMMISSION_PCT,0))	
1	0.0425

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数とNULL値

すべてのグループ関数は列内のNULL値を無視します。

ただし、NVL関数は、グループ関数でNULL値が含まれるようにします。

例:

1. COMMISSION_PCT列に有効な値が格納されている表内の行のみを対象として平均が計算されています。また、すべての従業員に支払われる歩合の合計が、歩合を受け取る従業員の数(4)で除算されています。
2. COMMISSION_PCT列にNULL値が格納されているかどうかに関係なく、表内のすべての行を対象として平均が計算されています。また、すべての従業員に支払われる歩合の合計が、社内の従業員の総数(20)で除算されています。

章の講義項目

- グループ関数:
 - 種類と構文
 - AVG、SUM、MIN、MAX、COUNTの使用
 - グループ関数内でのDISTINCTキーワードの使用
 - グループ関数内のNULL値
- 行のグループ化:
 - GROUP BY句
 - HAVING句
- グループ関数のネスト

ORACLE

Copyright © 2007, Oracle. All rights reserved.

データのグループの作成

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	5800
5	50	2500
6	50	2600
7	50	3100
8	50	3500
9	60	4200
10	60	6000
11	60	9000
12	80	11000
13	80	10500
14	80	8600
...		
19	110	12000
20	(null)	7000

4400

9500

3500

6400

10033

EMPLOYEE表内の 各部門の 平均給与

	DEPARTMENT_ID	AVG(SALARY)
1	10	4400
2	20	9500
3	50	3500
4	60	6400
5	80	10033.333333333333...
6	90	19333.333333333333...
7	110	10150
8	(null)	7000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

データのグループの作成

これまでのすべてのグループ関数では、表を1つの大きい情報グループとして扱ってきました。しかし、情報の表を小さいグループに分ける必要があります場合があります。これは、GROUP BY句を使用して実行できます。

データのグループの作成: GROUP BY句の構文

```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

GROUP BY句を使用して、表内の行を小さいグループに分けることができます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

データのグループの作成: GROUP BY句の構文

GROUP BY句を使用して、表内の行をグループに分けることができます。これによって、グループ関数を使用して、それぞれのグループのサマリー情報を戻すことができるようになります。

構文の説明

group_by_expression 行のグループ化の基準となる値を持つ列を指定します。

ガイドライン

- グループ関数をSELECT句に含めた場合、個々の列がGROUP BY句に表示されていない場合は、個々の結果も選択できません。GROUP BY句に列リストが含まれていない場合は、エラー・メッセージが表示されます。
- WHERE句を使用して、グループに分ける前行を除外できます。
- GROUP BY句には、列を含める必要があります。
- GROUP BY句では、列別名を使用できません。

GROUP BY句の使用法

グループ関数ではないSELECTリスト内のすべての列は、GROUP BY句に含める必要があります。

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id ;
```

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	90	19333.3333333333...
3	20	9500
4	110	10150
5	50	3500
6	80	10033.3333333333...
7	60	6400
8	10	4400

ORACLE

Copyright © 2007, Oracle. All rights reserved.

GROUP BY句の使用法

GROUP BY句を使用するときは、グループ関数ではないSELECTリスト内のすべての列をGROUP BY句に含めてください。スライドの例では、それぞれの部門の部門番号と平均給与が表示されています。GROUP BY句が含まれているこのSELECT文は、次のように計算されます。

- SELECT句によって、取得する列が次のように指定されています。
 - EMPLOYEES表の部門番号の列
 - GROUP BY句で指定したグループ内のすべての給与の平均
- FROM句によって、データベースがアクセスする表(EMPLOYEES表)が指定されています。
- WHERE句によって、取得する行が指定されます。WHERE句が存在しないため、デフォルトによりすべての行が取得されます。
- GROUP BY句によって、行をどのようにグループ化するかが指定されます。行は部門番号によってグループ化されているため、給与の列に適用されたAVG関数によって、それぞれの部門の平均給与が計算されます。

GROUP BY句の使用方法

SELECTリストにGROUP BY列を指定する必要はありません。

```
SELECT  AVG(salary)
FROM    employees
GROUP BY department_id ;
```

	AVG(SALARY)
1	7000
2	19333.3333333333333333333333333333...
3	9500
4	10150
5	3500
6	10033.3333333333333333333333333333...
7	6400
8	4400

ORACLE

Copyright © 2007, Oracle. All rights reserved.

GROUP BY句の使用方法(続き)

SELECT句にGROUP BY列を指定する必要はありません。たとえば、スライドのSELECT文は、それぞれの部門番号を表示せずに、各部門の平均給与を表示します。ただし、部門番号が表示されていない場合は、結果を有効に利用できません。

ORDER BY句で次のようにグループ関数を使用することもできます。

```
SELECT  department_id, AVG(salary)
FROM    employees
GROUP BY department_id
ORDER BY AVG(salary);
```

	DEPARTMENT_ID	AVG(SALARY)
1	50	3500
2	10	4400
3	60	6400
...		
7	110	10150
8	90	19333.3333333333333333333333333333...

複数の列によるグループ化

EMPLOYEES

	DEPARTMENT_ID	JOB_ID	SALARY
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_MAN	5800
5	50	ST_CLERK	2500
6	50	ST_CLERK	2600
7	50	ST_CLERK	3100
8	50	ST_CLERK	3500
9	60	IT_PROG	4200
10	60	IT_PROG	6000
11	60	IT_PROG	9000
12	80	SA_REP	11000
13	80	SA_MAN	10500
14	80	SA_REP	8600
...			
19	110	AC_MGR	12000
20	(null)	SA_REP	7000

EMPLOYEES表の部門別に
グループ化された職務ごとの
給与を加算します。

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	11700
5	50	ST_MAN	5800
6	60	IT_PROG	19200
7	80	SA_MAN	10500
8	80	SA_REP	19600
9	90	AD_PRES	24000
10	90	AD_VP	34000
11	110	AC_ACCOUNT	8300
12	110	AC_MGR	12000
13	(null)	SA_REP	7000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数の列によるグループ化

グループ内のグループの結果を表示することが必要な場合があります。スライドには、部門ごとにそれぞれの職種に対して支払われる給与の合計を表示するレポートが表示されています。

EMPLOYEES表は、最初に部門番号によってグループ化されて、次にそのグループ内の職種によってグループ化されています。たとえば、部門50の4人の在庫管理者がグループ化され、グループ内のすべての在庫管理者に対して1つの結果(合計給与)が生成されています。

複数の列に対するGROUP BY句の使用方法

```
SELECT    department_id dept_id, job_id, SUM(salary)
FROM      employees
GROUP BY  department_id, job_id
ORDER BY  department_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	11700
5	50	ST_MAN	5800
6	60	IT_PROG	19200
7	80	SA_MAN	10500
8	80	SA_REP	19600
9	90	AD PRES	24000
10	90	AD_VP	34000
11	110	AC_ACCOUNT	8300
12	110	AC_MGR	12000
13	(null)	SA_REP	7000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数の列に対するGroup By句の使用方法

複数のGROUP BY列をリストして、グループとサブグループのサマリー結果を戻すことができます。結果のデフォルトのソート順序は、GROUP BY句内の列の順序によって決まります。スライドの例では、GROUP BY句が含まれているSELECT文が次のように計算されます。

- SELECT句によって、取得する列が次のように指定されています。
 - EMPLOYEES表の部門番号
 - EMPLOYEES表の職務ID
 - GROUP BY句で指定したグループ内のすべての給与の合計
- FROM句によって、データベースがアクセスする表(EMPLOYEES表)が指定されています。
- GROUP BY句によって、行をグループ化する方法が次のように指定されています。
 - 最初に、部門番号で行をグループ化する。
 - 次に、部門番号グループの職務IDで行をグループ化する。

これによって、各部門番号グループのすべての職務IDに対する給与の列にSUM関数が適用されます。

グループ関数を使用した無効な問合せ

集計関数ではないSELECTリスト内の列または式は、GROUP BY句に含める必要があります。

```
SELECT department_id, COUNT(last_name)
FROM employees;
```

ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"

各department_idの姓をカウントするには、GROUP BY句を追加する必要があります。

```
SELECT department_id, job_id, COUNT(last_name)
FROM employees
GROUP BY department_id;
```

ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"

GROUP BYにjob_idを追加するか、SELECTリストからjob_id列を削除します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数を使用した無効な問合せ

個々の項目 (DEPARTMENT_ID) とグループ関数 (COUNT) を同じSELECT文に混在させる場合は、個々の項目 (この場合は、DEPARTMENT_ID) を指定するGROUP BY句を含める必要があります。GROUP BY句が省略されていると、エラー・メッセージ「は単一グループのグループ関数ではありません。」が表示されて、アスタリスク(*)によって問題のある列が示されます。スライドの最初の例では、GROUP BY句を追加してエラーを修正できます。

```
SELECT department_id, count(last_name)
FROM employees
GROUP BY department_id;
```

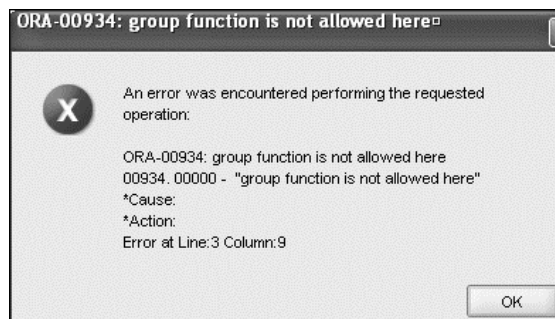
集計関数ではないSELECTリスト内の列または式はすべてGROUP BY句に含める必要があります。スライドの2番目の例では、job_idはGROUP BY句に含まれておらず、グループ関数でも使用されていないため、「はGROUP BY式ではありません。」というエラーが表示されています。2番目のスライドの例では、GROUP BY句にjob_idを追加してエラーを修正できます。

```
SELECT department_id, job_id, COUNT(last_name)
FROM employees
GROUP BY department_id, job_id;
```

グループ関数を使用した無効な問合せ

- WHERE句を使用して、グループを制限することはできません。
- グループを制限するには、HAVING句を使用します。
- WHERE句では、グループ関数を使用できません。

```
SELECT    department_id, AVG(salary)
FROM      employees
WHERE     AVG(salary) > 8000
GROUP BY  department_id;
```



WHERE句では
グループを
制限できない

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数を使用した無効な問合せ(続き)

WHERE句を使用してグループを制限することはできません。スライドの例に示すSELECT文は、平均給与が\$8,000を超える部門の平均給与の表示をWHERE句で制限しているため、エラーになります。

ただし、この例では、次のようにHAVING句を使用してグループを制限することでエラーを修正できます。

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY  department_id
HAVING    AVG(salary) > 8000;
```

	DEPARTMENT_ID	AVG(SALARY)
1	90	19333.333333333333...
2	20	9500
3	110	10150
4	80	10033.333333333333...

グループの結果の制限

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	5800
5	50	2500
6	50	2600
7	50	3100
8	50	3500
9	60	4200
10	60	6000
11	60	9000
12	80	11000
13	80	10500
14	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

最大給与が\$10,000を
超えている部門の
最大給与

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	80	11000
3	90	24000
4	110	12000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループの結果の制限

WHERE句を使用して選択する行を制限する場合と同じ方法で、HAVING句を使用してグループを制限します。最大給与が\$10,000を超えるそれぞれの部門の最大給与を求めるには、次の手順を実行する必要があります。

1. 部門番号でグループ化してそれぞれの部門の平均給与を検索します。
2. 最大給与が\$10,000を超える部門にグループを制限します。

HAVING句を使用したグループの結果の制限

HAVING句を使用すると、Oracleサーバーによって次のようにグループが制限されます。

1. 行がグループ化されます。
2. グループ関数が適用されます。
3. HAVING句に一致するグループが表示されます。

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

HAVING句によるグループ結果の制限

HAVING句を使用して表示するグループを指定することで、集計情報に基づいてグループがさらに制限されます。

この構文では、*group_condition*によって、指定された条件がTRUEになるグループに返される行のグループが制限されています。

Oracleサーバーでは、HAVING句を使用するときに次の手順が実行されます。

1. 行がグループ化されます。
2. グループ関数がグループに適用されます。
3. HAVING句内の基準に一致するグループが表示されます。

HAVING句はGROUP BY句の前に指定できますが、GROUP BY句を最初に指定して、より論理的にすることをお勧めします。グループの構成とグループ関数の計算が行われてから、SELECTリスト内のグループにHAVING句が適用されます。

HAVING句の使用方法

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY  department_id
HAVING    MAX(salary)>10000 ;
```

	DEPARTMENT_ID	MAX(SALARY)
1	90	24000
2	20	13000
3	110	12000
4	80	11000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

HAVING句の使用方法

スライドの例では、最大給与が\$10,000を超える部門の部門番号と最大給与が表示されています。SELECTリストでグループ関数を使用せずに、GROUP BY句を使用できます。グループ関数の結果に基づいて行を制限する場合は、HAVING句とともにGROUP BY句を使用する必要があります。次の例では、最大給与が\$10,000を超える部門の部門番号と平均給与が表示されます。

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY  department_id
HAVING    max(salary)>10000;
```

	DEPARTMENT_ID	AVG(SALARY)
1	90	19333.333333333333...
2	20	9500
3	110	10150
4	80	10033.333333333333...

HAVING句の使用方法

```
SELECT  job_id, SUM(salary) PAYROLL
FROM    employees
WHERE   job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING  SUM(salary) > 13000
ORDER BY SUM(salary);
```

	JOB_ID	PAYROLL
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

HAVING句の使用方法(続き)

スライドの例では、合計給与が\$13,000を超える各職務の職務IDと月給の合計が表示されています。この例では、販売担当者が除外され、月給の合計でリストがソートされています。

章の講義項目

- グループ関数:
 - 種類と構文
 - AVG、SUM、MIN、MAX、COUNTの使用
 - グループ関数内でのDISTINCTキーワードの使用
 - グループ関数内のNULL値
- 行のグループ化:
 - GROUP BY句
 - HAVING句
- グループ関数のネスト

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数のネスト

最大平均給与を表示します。

```
SELECT MAX(AVG(salary))  
FROM employees  
GROUP BY department_id;
```

	MAX(AVG(SALARY))
1	19333.3333333333333333333333333333

ORACLE

Copyright © 2007, Oracle. All rights reserved.

グループ関数のネスト

グループ関数は、2つの関数の深さまでネストできます。スライドの例では、department_idごとの平均給与が計算されて、最大平均給与が表示されています。

グループ関数をネストするときはGROUP BY句を使用する必要があることに注意してください。

まとめ

この章では、次のことを学習しました。

- グループ関数COUNT、MAX、MIN、SUMおよびAVGの使用
- ORDER BY句を使用する問合せの記述
- HAVING句を使用する問合せの記述

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

SQLでは、次のようなグループ関数を使用できます。

AVG、COUNT、MAX、MIN、SUM、STDDEVおよびVARIANCE

GROUP BY句を使用して、サブグループを作成できます。また、グループはHAVING句を使用して制限できます。

文では、HAVING句とGROUP BY句はWHERE句の後に指定します。WHERE句の後のGROUP BYおよびHAVING句の順序は重要ではありません。ORDER BY句は最後に指定します。

Oracleサーバーでは、次の順序で句が計算されます。

1. 文にWHERE句が含まれている場合は、サーバーによって対象の行が設定されます。
2. サーバーによって、GROUP BY句で指定されているグループが識別されます。
3. HAVING句によって、HAVING句のグループ基準を満たさないグループの結果がさらに制限されます。

注意: グループ関数の全リストについては、『Oracle Database SQL言語リファレンス11gリリース1 (11.1)』を参照してください。

演習5: 概要

この演習では次の項目について説明しています。

- グループ関数を使用する問合せの記述
- 複数の結果を求めるための行のグループ化
- HAVING句を使用したグループの制限

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習5: 概要

この演習を終えると、グループ関数の使用方法とデータのグループの選択方法を把握できるようになります。

演習5

次の3つの文が正しいかどうかを判断してください。○または×を円で囲みます。

1. グループ関数は複数の行に対して動作し、グループごとに1つの結果を返します。
○/×
2. グループ関数では計算にNULLが含まれます。
○/×
3. WHERE句は、グループ計算に取り込む前に行を制限します。
○/×

HR部門は次のレポートを必要としています。

4. すべての従業員の給与の最高額、最低額、合計および平均を求めます。列にそれぞれ、Maximum、Minimum、Sum、Averageというラベルを付けます。結果を最も近い整数に丸めます。作成したSQL文をlab_05_04.sqlとして保存します。問合せを実行します。



	Maximum	Minimum	Sum	Average
1	24000	2500	175500	8775

5. 職種ごとの給与の最低額、最高額、合計および平均を表示するように、lab_05_04.sqlの問合せを変更します。lab_05_04.sqlをlab_05_05.sqlとして保存し直します。lab_05_05.sqlの文を実行します。



	JOB_ID	Maximum	Minimum	Sum	Average
1	IT_PROG	9000	4200	19200	6400
2	AC_MGR	12000	12000	12000	12000
3	AC_ACCOUNT	8300	8300	8300	8300
4	ST_MAN	5800	5800	5800	5800
5	AD_ASST	4400	4400	4400	4400
6	AD_VP	17000	17000	34000	17000
7	SA_MAN	10500	10500	10500	10500
8	MK_MAN	13000	13000	13000	13000
9	AD PRES	24000	24000	24000	24000
10	SA_REP	11000	7000	26600	8867
11	MK_REP	6000	6000	6000	6000
12	ST_CLERK	3500	2500	11700	2925

演習5(続き)


- 同じ職務の従業員の数を表示する問合せを記述します。

	 JOB_ID	 COUNT(*)
1	AC_ACCOUNT	1
2	AC_MGR	1
3	AD_ASST	1
4	AD PRES	1
5	AD_VP	2
6	IT_PROG	3
7	MK_MAN	1
8	MK_REP	1
9	SA_MAN	1
10	SA_REP	3
11	ST_CLERK	4
12	ST_MAN	1

HR部門のユーザーに職種の入力を求めるように、問合せを汎用化します。スクリプトを lab_05_06.sql という名前のファイルに保存します。問合せを実行します。入力を求められたら、IT_PROG と入力します。

	 JOB_ID	 COUNT(*)
1	IT_PROG	3

- マネージャをリストせずにその数を特定します。列にNumber of Managers というラベルを付けます。ヒント: MANAGER_ID 列を使用して、マネージャの数を特定します。

	 Number of Managers
1	8

- 給与の最高額と最低額の差額を求めます。列にDIFFERENCE というラベルを付けます。

	 DIFFERENCE
1	21500

演習5(続き)

時間があるときは、次の演習問題に進みます。

- マネージャのマネージャ番号と、そのマネージャが管理する最も給与の低い従業員の給与を表示するレポートを作成します。マネージャが不明な従業員は除外します。また、給与の最低額が\$6,000以下のグループは除外します。出力を給与の降順でソートします。

	MANAGER_ID	MIN(SALARY)
1	102	9000
2	205	8300
3	149	7000

さらに演習を続ける場合は、次の演習問題に進みます。

- 従業員の総数と、1995年、1996年、1997年および1998年に雇用された従業員の数を表示する問合せを作成します。適切な列ヘッダーを作成します。

	TOTAL	1995	1996	1997	1998
1	20	1	2	2	3

- 部門20、50、80および100に対して、職務、部門番号に基づくその職務の給与、およびその職務の給与の合計を表示するマトリクス形式の問合せを作成して、それぞれの列に適切なヘッダーを付けます。

	Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
1	IT_PROG	(null)	(null)	(null)	(null)	19200
2	AC_MGR	(null)	(null)	(null)	(null)	12000
3	AC_ACCOUNT	(null)	(null)	(null)	(null)	8300
4	ST_MAN	(null)	5800	(null)	(null)	5800
5	AD_ASST	(null)	(null)	(null)	(null)	4400
6	AD_VP	(null)	(null)	(null)	34000	34000
7	SA_MAN	(null)	(null)	10500	(null)	10500
8	MK_MAN	13000	(null)	(null)	(null)	13000
9	AD_PRES	(null)	(null)	(null)	24000	24000
10	SA_REP	(null)	(null)	19600	(null)	26600
11	MK_REP	6000	(null)	(null)	(null)	6000
12	ST_CLERK	(null)	11700	(null)	(null)	11700

Oracle Internal & Oracle Academy
Use Only

複数の表のデータの表示

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

この章を終えると、次のことができるようになります。

- 等価結合および非等価結合を使用した、複数の表からデータにアクセスするSELECT文の記述
- 自己結合による表自体の結合
- 通常は結合条件を満たさないデータの外部結合での表示
- 2つ以上の表のすべての行のデカルト積の生成

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

この章では、複数の表のデータを取得する方法について説明します。結合は、複数の表の情報を表示するために使用されます。したがって、表を結合することにより、複数の表の情報を表示することができます。

注意: 結合については、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のSQL問合せおよび副問合せの結合に関する節を参照してください。

章の講義項目

- 結合(JOIN)のタイプと構文
- 自然結合:
 - USING句
 - ON句
- 自己結合
- 非等価結合
- 外部(OUTER)結合:
 - 左側外部(LEFT OUTER)結合
 - 右側外部(RIGHT OUTER)結合
 - 完全外部(FULL OUTER)結合
- デカルト積:
 - クロス結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数の表からのデータの取得

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
...			
18	202	Fay	20
19	205	Higgins	110
20	206	Gietz	110

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
5	144	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数の表からのデータの取得

複数の表のデータを使用する必要がある場合があります。スライドの例では、2つの別の表のデータがレポートに表示されています。

- 従業員IDはEMPLOYEES表にあります。
- 部門IDはEMPLOYEES表とDEPARTMENTS表の両方に表示されています。
- 部門名はDEPARTMENTS表に表示されています。

レポートを作成するには、EMPLOYEES表とDEPARTMENTS表をリンクし、両方の表のデータにアクセスする必要があります。

結合のタイプ

SQL:1999規格に準拠する結合には、次のタイプがあります。

- 自然結合:
 - NATURAL JOIN句
 - USING句
 - ON句
- 外部結合:
 - 左側外部結合 (LEFT OUTER JOIN)
 - 右側外部結合 (RIGHT OUTER JOIN)
 - 完全外部結合 (FULL OUTER JOIN)
- クロス結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

結合のタイプ

表の結合には、SQL:1999規格に準拠した結合構文を使用できます。

注意: Oracle9より前のリリースでは、結合構文は米国規格協会 (ANSI) 標準と異なっていました。SQL:1999準拠の結合構文には、以前のリリースで使用されていたOracle独自の結合構文と比べ、パフォーマンス上の利点はありません。独自の結合構文の詳細は、付録C「Oracle結合構文」を参照してください。

注意: SQL:1999の結合構文については、次のスライドで説明します。

SQL:1999構文を使用した表の結合

複数の表のデータを問い合わせるには、次のように結合を使用します。

```
SELECT    table1.column, table2.column
FROM      table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SQL:1999構文を使用した表の結合

構文の説明

*table1.column*は、データの取得元の表と列を示します。

NATURAL JOINは、同じ列名に基づいて2つの表を結合します。

JOIN *table2* USING *column_name*は、列名に基づいて等価結合を実行します。

JOIN *table2* ON *table1.column_name* = *table2.column_name*は、ON句の条件に基づいて等価結合を実行します。

LEFT/RIGHT/FULL OUTERは、外部結合の実行に使用されます。

CROSS JOINは、2つの表からデカルト積を戻します。

詳細は、『Oracle Database SQL言語リファレンス11gリリース1(11.1)』のSELECTに関する節を参照してください。

あいまいな列名の修飾

- 複数の表にある列名を修飾するには、表接頭辞を使用します。
- 表接頭辞を使用すると、パフォーマンスが向上します。
- 完全な名前の表接頭辞ではなく、表別名を使用します。
- 表別名は表の短い名前です。
 - SQLコードが短くなるため、メモリーの使用量が削減される
- 異なる表にある同一名の列を区別するには、列別名を使用します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

あいまいな列名の修飾

2つ以上の表を結合する場合、表の名前で列名を修飾して、あいまいさを回避する必要があります。表接頭辞を使用しない場合、SELECTリストのDEPARTMENT_ID列は、DEPARTMENTS表またはEMPLOYEES表のいずれからでも取得できます。表接頭辞を追加して問合せを実行する必要があります。2つの表に共通の列名がない場合には、列の修飾は不要です。ただし、表接頭辞を使用すると、検索する列の正確な場所がOracleサーバーによって認識されるため、パフォーマンスが向上します。

しかし、表の名前による列名の修飾には時間がかかることがあります。表名が長い場合には特に時間がかかります。表明を使用する代わりに、表別名を使用します。列別名が列の別名であると同様に、表別名は表の別名です。表別名を使用することにより、SQLコードが短くなるため、メモリーの使用量を削減できます。

表の名前には、完全な名前を指定し、その後に空白と表別名を続けます。たとえば、EMPLOYEES表には別名eを指定し、DEPARTMENTS表には別名dを指定できます。

ガイドライン

- 表別名には30文字まで指定できますが、別名は短い方が便利です。
- FROM句で特定の表名に対して表別名を使用した場合、SELECT文では表名の代わりにその表別名を使用する必要があります。
- 表別名には、わかりやすい名前を指定する必要があります。
- 表別名は、現在のSELECT文に対してのみ有効です。

章の講義項目

- 結合 (JOIN) の種類と構文
- 自然結合:
 - USING句
 - ON句
- 自己結合
- 非等価結合
- 外部 (OUTER) 結合:
 - 左側外部 (LEFT OUTER) 結合
 - 右側外部 (RIGHT OUTER) 結合
 - 完全外部 (FULL OUTER) 結合
- デカルト積:
 - クロス結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

自然結合の作成

- NATURAL JOIN句は、2つの表にある同じ名前を持つすべての列に基づきます。
- 一致するすべての列で値が等しい行が2つの表から選択されます。
- 同じ名前の列でデータ型が異なる場合、エラーが戻されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

自然結合の作成

2つの表でデータ型と名前が一致する列を対象にして自動的に表を結合できます。この場合、NATURAL JOINキーワードを使用します。

注意: 両方の表で名前とデータ型が一致する列のみを結合できます。列の名前が同じでも、データ型が異なる場合、NATURAL JOIN構文でエラーが発生します。

自然結合によるレコードの取得

```
SELECT department_id, department_name,  
       location_id, city  
FROM   departments  
NATURAL JOIN locations ;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

ORACLE

Copyright © 2007, Oracle. All rights reserved.

自然結合によるレコードの取得

スライドの例では、LOCATIONS表は、LOCATION_ID列によってDEPARTMENT表に結合されています。この列は両方の表で同じ名前を持つ唯一の列です。共通する別の列が存在する場合には、そのすべて列が結合に使用されます。

WHERE句による自然結合

WHERE句を使用した自然結合では、別の制約を実装できます。次の例では、出力の行を部門IDが20または50の行に限定しています。

```
SELECT department_id, department_name,  
       location_id, city  
FROM   departments  
NATURAL JOIN locations  
WHERE  department_id IN (20, 50);
```

USING句による結合の作成

- 名前は同じでもデータ型が異なる列が複数ある場合、USING句を使用して自然結合を適用し、等価結合に使用する列を指定することができます。
- USING句は、複数の列が一致するときに1つの列のみを一致させる場合に使用します。
- NATURAL JOIN句とUSING句は、相互に排他的です。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

USING句による結合の作成

自然結合では、名前とデータ型が一致するすべての列が表の結合に使用されます。等価結合に使用する列のみを指定するには、USING句を使用します。

列名の結合

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
100	90
101	90
102	90
103	60
104	60
107	60
124	50
141	50
142	50
143	50
144	50
149	80
174	80
176	80

...

外部キー

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
1	10 Administration
2	20 Marketing
3	50 Shipping
4	60 IT
5	80 Sales
6	90 Executive
7	110 Accounting
8	190 Contracting

主キー

ORACLE

Copyright © 2007, Oracle. All rights reserved.

列名の結合

従業員の部門名を確認するには、EMPLOYEES表のDEPARTMENT_ID列の値とDEPARTMENTS表のDEPARTMENT_IDの値を比較します。EMPLOYEES表とDEPARTMENTS表の関係は等価結合です。つまり、両方の表のDEPARTMENT_ID列に同じ値が使用されている必要があります。通常、このタイプの結合には、主キーと外部キーの補集合が使用されます。

注意: 等価結合は、単純結合または内部結合とも呼ばれます。

USING句によるレコードの取得

```
SELECT employee_id, last_name,  
       location_id, department_id  
FROM   employees JOIN departments  
USING (department_id) ;
```

	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	124	Mourgos	1500	50
5	144	Vargas	1500	50
6	143	Matos	1500	50
7	142	Davies	1500	50
8	141	Rajs	1500	50
9	107	Lorentz	1400	60
10	104	Ernst	1400	60
...				
19	205	Higgins	1700	110

ORACLE

Copyright © 2007, Oracle. All rights reserved.

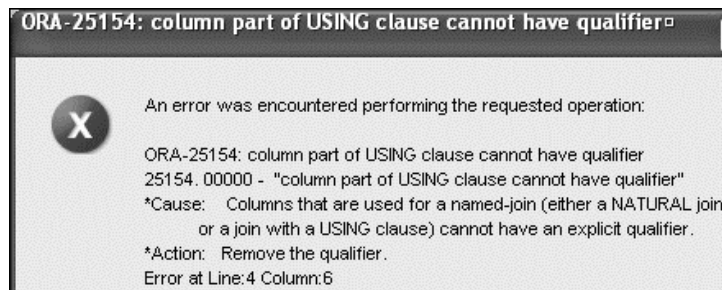
USING句によるレコードの取得

スライドの例では、EMPLOYEES表とDEPARTMENTS表のDEPARTMENT_ID列が結合され、従業員が勤務する部門のLOCATION_IDが表示されています。

USING句での表別名の使用方法

- USING句で使用されている列は修飾しないでください。
- 同じ列がSQL文の他の場所で使用されている場合、その列に別名を使用しないでください。

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d
USING (location_id)
WHERE  d.location_id = 1400;
```



ORACLE

Copyright © 2007, Oracle. All rights reserved.

USING句での表別名の使用方法

USING句で結合する場合、USING句自体で使用されている列を修飾することはできません。また、その列がSQL文の別の場所で使用されている場合には、その列に別名を使用できません。たとえば、スライドの間合せでは、location_id列はUSING句で使用されているため、WHERE句のこの列には別名を使用できません。

USING句で参照される列には、SQL文のどの場所にも修飾子(表名または別名)を使用できません。たとえば、次の文は有効です。

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE  location_id = 1400;
```

両方の表に共通しており、USING句で使用されていない他の列は、列名の前に表別名を付加する必要があります。付加しない場合は、「列の定義が未確定です。」というエラーが表示されます。

次の文において、manager_idがemployees表とdepartments表の両方にあり、manager_idの前に表別名が付加されていない場合、「列の定義が未確定です。」というエラーが表示されます。

次の文は有効です。

```
SELECT first_name, d.department_name, d.manager_id
FROM   employees e JOIN departments d USING (department_id)
WHERE  department_id = 50;
```

ON句による結合の作成

- 自然結合の結合条件は、基本的には同じ名前を持つすべての列の等価結合です。
- 任意の条件を指定する場合や結合する列を指定する場合には、ON句を使用します。
- 結合条件は、他の検索条件とは独立しています。
- ON句を使用すると、コードがわかりやすくなります。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ON句による結合の作成

結合条件を指定するには、ON句を使用します。この場合、WHERE句の検索条件やフィルタ条件から独立した結合条件を指定できます。

ON句によるレコードの取得

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON      (e.department_id = d.department_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	1400
10	104	Ernst	60	60	1400

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ON句によるレコードの取得

この例では、EMPLOYEES表とDEPARTMENTS表のDEPARTMENT_ID列がON句によって結合されています。EMPLOYEES表の部門IDがDEPARTMENTS表の部門IDと等しい場合に、行が戻されます。一致したcolumn_namesを修飾するには表別名を使用する必要があります。

また、名前が異なる列の結合にもON句が使用できます。スライドの例にある「(e.department_id = d.department_id)」のような、結合する列を囲むカッコは省略することができます。したがって、「ON e.department_id = d.department_id」と指定しても動作します。

注意: 2つのdepartment_idを区別するために、接尾辞「_1」がSQL Developerによって追加されています。

ON句による3方向結合の作成

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ON句による3方向結合の作成

3方向結合とは、3つの表を結合することです。SQL:1999準拠構文においては、結合は左から右に実行されます。したがって、最初に行われる結合は、EMPLOYEES JOIN DEPARTMENTSです。最初の結合条件は、EMPLOYEESとDEPARTMENTSの列を参照できますが、LOCATIONSの列は参照できません。2番目の結合条件は、3つの表すべての列を参照できます。

注意: スライドのコード例は、次のようにUSING句を使用しても実行できます。

```
SELECT e.employee_id, l.city, d.department_name
FROM employees e
JOIN departments d
  USING (department_id)
JOIN locations l
  USING (location_id)
```

結合への追加条件の適用

追加条件を適用するには、AND句またはWHERE句を使用します。

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON      (e.department_id = d.department_id)  
AND     e.manager_id = 149 ;
```

または

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON      (e.department_id = d.department_id)  
WHERE   e.manager_id = 149 ;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

結合への追加条件の適用

結合に追加条件を適用できます。

例では、EMPLOYEES表とDEPARTMENTS表の結合が実行され、さらに、マネージャIDが149の従業員のみが表示されます。AND句を追加することで、ON句に追加条件を追加できます。また、追加条件の適用にはWHERE句も使用できます。

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	174	Abel	80	80	2500
2	176	Taylor	80	80	2500

章の講義項目

- 結合 (JOIN) の種類と構文
- 自然結合:
 - USING句
 - ON句
- 自己結合
- 非等価結合
- 外部 (OUTER) 結合:
 - 左側外部 (LEFT OUTER) 結合
 - 右側外部 (RIGHT OUTER) 結合
 - 完全外部 (FULL OUTER) 結合
- デカルト積:
 - クロス結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

表自体の結合

EMPLOYEES(WORKER)

	EMPLOYEE_ID	LAST_NAME	MANAGER_ID
1	100	King	(null)
2	101	Kochhar	100
3	102	De Haan	100
4	103	Hunold	102
5	104	Ernst	103
6	107	Lorentz	103
7	124	Mourgos	100
8	141	Rajs	124
9	142	Davies	124
10	143	Matos	124

...

EMPLOYEES(MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos
141	Rajs
142	Davies
143	Matos

...

↑
WORKER表のMANAGER_IDは、
MANAGER表のEMPLOYEE_IDと等しい。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

表自体の結合

表をその表自体に結合する必要がある場合があります。各従業員のマネージャの名前を検索するには、EMPLOYEES表をその表自体に結合する(すなわち、自己結合を実行する)必要があります。たとえば、Lorentzのマネージャの名前を検索するには、次の手順を実行する必要があります。

- EMPLOYEES表のLAST_NAME列でLorentzを検索します。
- MANAGER_ID列でLorentzのマネージャ番号を確認します。Lorentzのマネージャ番号は103です。
- LAST_NAME列でEMPLOYEE_IDが103のマネージャの名前を検索します。Hunoldの従業員番号が103です。したがって、HunoldがLorentzのマネージャとなります。

この手順では、表を2回検索します。1回目は、LAST_NAME列でLorentzを検索し、MANAGER_IDの値(103)を確認しています。2回目は、EMPLOYEE_ID列で103を検索し、LAST_NAME列でHunoldを確認しています。

ON句による自己結合の作成

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON     (worker.manager_id = manager.employee_id);
```

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
8	Kochhar	King
9	Hartstein	King
10	De Haan	King

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

ON句による自己結合の作成

ON句は、同じ表内または異なる表で、異なる名前を持つ列を結合する場合にも使用できます。例では、EMPLOYEE_ID列とMANAGER_ID列に基づくEMPLOYEES表の自己結合が示されています。

注意: スライドの例にある「(e.manager_id = m.employee_id)」のような、結合する列を囲むカッコは省略することができます。したがって、「ON e.manager_id = m.employee_id」と指定しても動作します。

章の講義項目

- 結合 (JOIN) の種類と構文
- 自然結合:
 - USING句
 - ON句
- 自己結合
- 非等価結合
- 外部 (OUTER) 結合:
 - 左側外部 (LEFT OUTER) 結合
 - 右側外部 (RIGHT OUTER) 結合
 - 完全外部 (FULL OUTER) 結合
- デカルト積:
 - クロス結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

非等価結合

EMPLOYEES

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hunold	9000
5	Ernst	6000
6	Lorentz	4200
7	Mourgos	5800
8	Rajs	3500
9	Davies	3100
10	Matos	2600
...		
19	Higgins	12000
20	Gietz	8300

JOB_GRADES

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

JOB_GRADES表によって、それぞれのGRADE_LEVELに対応するLOWEST_SALとHIGHEST_SALの値の範囲が定義されます。したがって、GRADE_LEVEL列を使用して、それぞれの従業員に等級を割り当てることができます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

非等価結合

非等価結合は、等価演算子ではない演算子による結合条件です。

EMPLOYEES表とJOB_GRADES表の関係は、非等価結合となります。EMPLOYEES表のSALARY列の値は、JOB_GRADES表のLOWEST_SAL列とHIGHEST_SAL列の値の範囲内にあります。したがって、各従業員を給与に基づいて等級分けすることができます。この関係の取得には、等価演算子(=)ではない演算子を使用されます。

非等価結合によるレコードの取得

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
      BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C

...

ORACLE

Copyright © 2007, Oracle. All rights reserved.

非等価結合によるレコードの取得

スライドの例では、従業員の給与等級を評価するために非等価結合が作成されています。給与は、下限と上限のいずれかの組合せの範囲内に必要があります。

この問合せを実行したときに、すべての従業員が一度だけしか表示されないことに注意してください。従業員が重複してリストに表示されることはありません。これには、次の2つの理由があります。

- JOB_GRADES表には、等級が重複する行は含まれません。すなわち、ある従業員の給与値は、給与等級表のいずれか1つの行の最高給与と最低給与の組合せの範囲内にあります。
- すべての従業員の給与は、職務等級表で設定されている範囲内にあります。すなわち、給与がLOWEST_SAL列の最低値よりも低い従業員や、HIGHEST_SAL列の最高値を超える従業員は存在しません。

注意: 別の条件(<= や >= など)を使用することもできますが、BETWEENは最も簡単に使用できます。BETWEEN条件を使用する際には、最初に下限値を指定し、次に上限値を指定してください。Oracleサーバーでは、BETWEEN条件はAND条件の組合せに変換されます。したがって、BETWEEN条件を使用してもパフォーマンス上の利点はありませんが、論理的に使用できるため操作は簡単です。

スライドの例では、あいまいさを回避するためではなく、パフォーマンス上の理由から表別名が指定されています。

章の講義項目

- 結合 (JOIN) の種類と構文
- 自然結合:
 - USING句
 - ON句
- 自己結合
- 非等価結合
- 外部 (OUTER) 結合:
 - 左側外部 (LEFT OUTER) 結合
 - 右側外部 (RIGHT OUTER) 結合
 - 完全外部 (FULL OUTER) 結合
- デカルト積:
 - クロス結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

外部結合による直接一致しないレコードの取得

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	90	King
2	90	Kochhar
3	90	De Haan
4	60	Hunold
5	60	Ernst
6	60	Lorentz
7	50	Mourgos
8	50	Rajs
9	50	Davies
10	50	Matos
...		
19	110	Higgins
20	110	Gietz

部門190には従業員が存在しない。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

外部結合による直接一致しないレコードの取得

行が結合条件を満たしていない場合、その行は問合せ結果に表示されません。たとえば、EMPLOYEES表とDEPARTMENTS表の等価結合条件では、部門ID190の従業員はEMPLOYEES表に存在しないため、この部門IDは表示されません。したがって、結果には従業員20名分のレコードではなく、19名分のレコードが表示されています。

従業員の存在しない部門のレコードを戻すには、外部結合を使用します。

内部結合と外部結合

- SQL:1999では、一致した行のみを戻す2つの表の結合を内部結合といいます。
- 2つの表の結合において、内部結合の結果とともに、左側（または右側）の表の不一致行を戻すものを左側（または右側）外部結合といいます。
- 2つの表の結合において、内部結合の結果とともに、左側と右側の結合結果を戻すものを完全外部結合といいます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

内部結合と外部結合

NATURAL JOIN句、USING句またはON句を使用して表を結合すると、内部結合になります。一致しない行は、出力に表示されません。一致しない行を戻すには、外部結合を使用します。外部結合では、結合条件を満たす行がすべて戻され、一方の表に結合条件を満たす行が存在しない場合に、もう一方の表の行の一部または全部を返します。

外部結合には、次の3つのタイプがあります。

- 左側外部 (LEFT OUTER)
- 右側外部 (RIGHT OUTER)
- 完全外部 (FULL OUTER)

左側外部結合 (LEFT OUTER JOIN)

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping
...			
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

ORACLE

Copyright © 2007, Oracle. All rights reserved.

左側外部結合 (LEFT OUTER JOIN)

この問合せでは、DEPARTMENTS表に一致する行が存在しない場合にも、左側の表であるEMPLOYEES表のすべての行が取得されます。

右側外部結合 (RIGHT OUTER JOIN)

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting
...			
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

ORACLE

Copyright © 2007, Oracle. All rights reserved.

右側外部結合 (RIGHT OUTER JOIN)

この問合せでは、EMPLOYEES表に一致する行が存在しない場合にも、右側の表であるDEPARTMENTS表のすべての行が取得されます。

完全外部結合 (FULL OUTER JOIN)

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

...

19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

ORACLE

Copyright © 2007, Oracle. All rights reserved.

完全外部結合 (FULL OUTER JOIN)

この問合せでは、DEPARTMENTS表に一致する行が存在しない場合にも、EMPLOYEES表のすべての行が取得されます。また、EMPLOYEES表に一致する行が存在しない場合にも、DEPARTMENTS表のすべての行が取得されます。

章の講義項目

- 結合 (JOIN) の種類と構文
- 自然結合:
 - USING句
 - ON句
- 自己結合
- 非等価結合
- 外部 (OUTER) 結合:
 - 左側外部 (LEFT OUTER) 結合
 - 右側外部 (RIGHT OUTER) 結合
 - 完全外部 (FULL OUTER) 結合
- デカルト積:
 - クロス結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

デカルト積

- デカルト積は次の場合に生成されます。
 - 結合条件が省略されている場合
 - 結合条件が無効な場合
 - 1つ目の表のすべての行が2つ目の表のすべての行に結合されている場合
- デカルト積を回避するには、有効な結合条件が常に含まれるようにします。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

デカルト積

結合条件が無効な場合や完全に省略されている場合、デカルト積が生成され、行のすべての組合せが表示されます。1つ目の表のすべての行が2つ目の表のすべての行に結合されます。

デカルト積では膨大な数の行が生成されることが多く、その結果はあまり有効ではありません。したがって、すべての表のすべての行を組み合わせる必要が特にないかぎり、必ず有効な結合条件を指定してください。

一方、十分な量のデータをシミュレートするために膨大な数の行を生成する必要があるテストでは、デカルト積が便利です。

デカルト積の生成

EMPLOYEES (20行)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100 King	90
2	101 Kochhar	90
3	102 De Haan	90
4	103 Hunold	60

...		
19	205 Higgins	110
20	206 Gietz	110

DEPARTMENTS (8行)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10 Administration	1700
2	20 Marketing	1800
3	50 Shipping	1500
4	60 IT	1400
5	80 Sales	2500
6	90 Executive	1700
7	110 Accounting	1700
8	190 Contracting	1700

デカルト積:
 $20 \times 8 = 160$ 行

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	100	90
2	101	90
3	102	90
4	103	60

...			
159	205	110	1700
160	206	110	1700

ORACLE

Copyright © 2007, Oracle. All rights reserved.

デカルト積の生成

結合条件が省略されている場合、デカルト積が生成されます。スライドの例では、EMPLOYEES表とDEPARTMENTS表の従業員の姓と部門名が表示されています。結合条件が指定されていないため、EMPLOYEES表のすべての行(20行)がDEPARTMENTS表のすべての行(8行)に結合され、その結果、出力には160の行が生成されます。

クロス結合の作成

- CROSS JOIN句は、2つの表のクロス積を作成します。
- クロス積は、2つの表のデカルト積とも呼ばれます。

```
SELECT last_name, department_name  
FROM   employees  
CROSS JOIN departments ;
```

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration
...		
159	Whalen	Contracting
160	Zlotkey	Contracting

ORACLE

Copyright © 2007, Oracle. All rights reserved.

クロス結合の作成

スライドの例では、EMPLOYEES表とDEPARTMENTS表のデカルト積が作成されています。

まとめ

この章では、次の結合を使用して複数の表のデータを表示する結合方法を学習しました。

- 等価結合
- 非等価結合
- 外部結合
- 自己結合
- クロス結合
- 自然結合
- 完全(両側)外部結合

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

表の結合には複数の方法があります。

結合のタイプ

- 等価結合
- 非等価結合
- 外部結合
- 自己結合
- クロス結合
- 自然結合
- 完全(両側)外部結合

デカルト積

デカルト積では、行のすべての組合せが表示されます。これは、WHERE句を省略するかCROSS JOIN句を指定すると実行されます。

表別名

- 表別名によりデータベースのアクセス速度が向上します。
- 表別名を使用すると、SQLコードを短くし、メモリーを節約することができます。
- 表別名は、列のあいまいさを回避するために必須となることもあります。

演習6: 概要

この演習では次の項目について説明しています。

- 等価結合を使用した表の結合
- 外部結合および自己結合の実行
- 条件の追加

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習6: 概要

この演習では、SQL:1999準拠の結合を使用して複数の表からデータを抽出します。

演習6

1. HR部門を検索して、すべての部門の住所を出力する問合せを記述します。LOCATIONS表とCOUNTRIES表を使用します。出力には、所在地ID、番地、市、州または県、および国を表示します。結果の出力には、NATURAL JOINを使用します。

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.	Toronto	Ontario	Canada
5	2500	Magdalen Centre, The ...	Oxford	Oxford	United Kingdom

2. HR部門はすべての従業員のレポートを必要としています。すべての従業員の姓、部門番号および部門名を表示する問合せを記述します。

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping
9	Hunold	60	IT
10	Ernst	60	IT

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting

演習6(続き)

- HR部門はトロントの従業員のレポートを必要としています。トロントで勤務しているすべての従業員の姓、職務、部門番号および部門名を表示します。

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

- 従業員の姓、従業員番号および担当マネージャの姓とマネージャ番号を表示するレポートを作成します。それぞれの列には、Employee、Emp#、ManagerおよびMgr#のラベルを付けます。作成したSQL文をlab_06_04.sqlとして保存します。問合せを実行します。

	Employee	EMP#	Manager	Mgr#
1	Kochhar	101	King	100
2	De Haan	102	King	100
3	Hunold	103	De Haan	102
4	Ernst	104	Hunold	103
5	Lorentz	107	Hunold	103
6	Mourgos	124	King	100
7	Rajs	141	Mourgos	124
8	Davies	142	Mourgos	124
9	Matos	143	Mourgos	124
10	Vargas	144	Mourgos	124

...

15	Whalen	200	Kochhar	101
16	Hartstein	201	King	100
17	Fay	202	Hartstein	201
18	Higgins	205	Kochhar	101
19	Gietz	206	Higgins	205

演習6(続き)

- 担当マネージャのいないKingを含め、すべての従業員が表示されるようにlab_06_04.sqlを変更します。結果は、従業員番号の順に並べます。作成したSQL文をlab_06_05.sqlとして保存します。lab_06_05.sqlの問合せを実行します。

	Employee	EMP#	Manager	Mgr#
1	King	100	(null)	(null)
2	Kochhar	101	King	100
3	De Haan	102	King	100
4	Hunold	103	De Haan	102
5	Ernst	104	Hunold	103
6	Lorentz	107	Hunold	103
7	Mourgos	124	King	100
8	Rajs	141	Mourgos	124
9	Davies	142	Mourgos	124
10	Matos	143	Mourgos	124

...

18	Fay	202	Hartstein	201
19	Higgins	205	Kochhar	101
20	Gietz	206	Higgins	205

- HR部門用のレポートを作成して、従業員の姓、部門番号、および指定した従業員と同じ部門に勤務するすべての従業員を表示します。それぞれの列に、該当するラベルを付けます。作成したスクリプトをlab_06_06.sqlという名前でファイルに保存します。

	DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20	Fay	Hartstein
2	20	Hartstein	Fay
3	50	Davies	Matos
4	50	Davies	Mourgos
5	50	Davies	Rajs
6	50	Davies	Vargas
7	50	Matos	Davies
8	50	Matos	Mourgos
9	50	Matos	Rajs
10	50	Matos	Vargas

...

42	110	Higgins	Gietz
----	-----	---------	-------

演習6(続き)

- HR部門では、職務等級と給与に関するレポートを必要としています。JOB_GRADES表を理解するために、まず、JOB_GRADES表の構造を確認します。次に、すべての従業員の名前、職務、部門名、給与および等級を表示する問合せを作成します。

DESC JOB_GRADES		
Name	Null	Type


GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER
3 rows selected		

	LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1	Vargas	ST_CLERK	Shipping	2500	A
2	Matos	ST_CLERK	Shipping	2600	A
3	Davies	ST_CLERK	Shipping	3100	B
4	Rajs	ST_CLERK	Shipping	3500	B
5	Lorentz	IT_PROG	IT	4200	B
6	Whalen	AD_ASST	Administration	4400	B
7	Mourgos	ST_MAN	Shipping	5800	B
8	Ernst	IT_PROG	IT	6000	C
9	Fay	MK_REP	Marketing	6000	C
10	Gietz	AC_ACCOUNT	Accounting	8300	C
...					
18	De Haan	AD_VP	Executive	17000	E
19	King	AD PRES	Executive	24000	E

演習6(続き)

さらに演習を続ける場合は、次の演習問題に進みます。

- HR部門では、Daviesより後に雇用されたすべての従業員の名前を調べています。従業員Daviesより後に雇用されたすべての従業員の名前と雇用日を表示する問合せを作成します。

	 LAST_NAME	HIRE_DATE
1	Lorentz	07-FEB-99
2	Mourgos	16-NOV-99
3	Matos	15-MAR-98
4	Vargas	09-JUL-98
5	Zlotkey	29-JAN-00
6	Taylor	24-MAR-98
7	Grant	24-MAY-99
8	Fay	17-AUG-97

- HR部門では、担当マネージャより前に雇用されたすべての従業員の名前と雇用日、およびその担当マネージャの名前と雇用日を調べる必要があります。作成したスクリプトをlab_c_09.sqlという名前でファイルに保存します。

	 LAST_NAME	HIRE_DATE	 LAST_NAME_1	HIRE_DATE_1
1	Whalen	17-SEP-87	Kochhar	21-SEP-89
2	Hunold	03-JAN-90	De Haan	13-JAN-93
3	Vargas	09-JUL-98	Mourgos	16-NOV-99
4	Matos	15-MAR-98	Mourgos	16-NOV-99
5	Davies	29-JAN-97	Mourgos	16-NOV-99
6	Rajs	17-OCT-95	Mourgos	16-NOV-99
7	Grant	24-MAY-99	Zlotkey	29-JAN-00
8	Taylor	24-MAR-98	Zlotkey	29-JAN-00
9	Abel	11-MAY-96	Zlotkey	29-JAN-00

Oracle Internal & Oracle Academy
Use Only

副問合せによる問合せの解決方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

目的

この章を終えると、次のことができるようになります。

- 副問合せの定義
- 副問合せで解決可能な問題の種類の説明
- 副問合せの種類の列挙
- 単一行副問合せと複数行副問合せの記述

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

この章では、SELECT文のさらに高度な機能について学習します。別のSQL文のWHERE句に副問合せを記述することにより、未知の条件値に基づいて値を得ることができます。また、この章では、単一行副問合せと複数行副問合せについても取り上げます。

章の講義項目

- 副問合せ: 種類、構文およびガイドライン
- 単一行副問合せ
 - 副問合せにおけるグループ関数
 - HAVING句での副問合せ
- 複数行副問合せ
 - ALL演算子またはANY演算子の使用
- 副問合せにおけるNULL値

ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せによる問題の解決方法

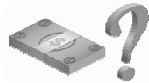
Abelより給与が高いのは誰か?

主問合せ:



Abelより給与が多い従業員は誰か?

副問合せ:



Abelの給与額は?



ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せによる問題の解決方法

Abelより給与の多い従業員を検索する問合せを記述するとします。

この問題を解決するには、2つの問合せが必要です。1つ目でAbelの給与額を検索し、2つ目の問合せで、その額より給与の多い従業員を検索します。

これら2つの問合せを組み合わせ、1つの問合せをもう一方の問合せの内部に配置することにより、この問題を解決することができます。

内側の問合せ(副問合せ)で戻された値は、外側の問合せ(主問合せ)で使用されます。副問合せを使用することは、2つの問合せを連続して実行し、最初の問合せの結果を2番目の問合せの検索値として使用することと同等です。

副問合せの構文

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT  select_list
         FROM    table);
```

- 副問合せ(内側の問合せ)は、主問合せ(外側の問合せ)より先に実行されます。
- 副問合せの結果は、主問合せに使用されます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せの構文

副問合せとは、別のSELECT文の句に埋め込まれているSELECT文のことです。副問合せを使用することにより、単純な文から高度な文を作成できます。副問合せは、表自体のデータに依存する条件によって表から行を選択する場合に、非常に有効です。

副問合せは、次の句をはじめ、様々なSQL句に配置できます。

- WHERE句
- HAVING句
- FROM句

構文の説明

*operator*には、比較条件(>、=、INなど)を使用できます。

注意: 比較条件は、単一行演算子(>、=、>=、<、<>、<=)と複数行演算子(IN、ANY、ALL)の2種類に分類されます。

副問合せは、ネストしたSELECT文、副SELECT文、または内側のSELECT文と呼ばれることもあります。通常は、副問合せが最初に実行され、その出力を使用して主問合せ(外側の問合せ)の問合せ条件が完成されます。

副問合せの使用方法

```
SELECT last_name, salary
FROM   employees 11000 ←
WHERE  salary >
      (SELECT salary
       FROM   employees
       WHERE  last_name = 'Abel');
```

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せの使用方法

スライドでは、内側の問合せによって従業員Abelの給与が確認されます。外側の問合せは、内側の問合せの結果を使用して、従業員Abelより給与の多い従業員をすべて表示します。

副問合せの使用に関するガイドライン

- 副問合せはカッコで囲みます。
- 読みやすくするため、副問合せは比較条件の右側に配置します（ただし、副問合せは比較演算子のどちら側にあっても問題ありません）。
- 単一行演算子は単一行副問合せで使用し、複数行演算子は複数行副問合せで使用します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せの使用に関するガイドライン

- 副問合せはカッコで囲む必要があります。
- 読みやすくするため、副問合せは比較条件の右側に配置します。ただし、副問合せは比較演算子のどちら側にあっても問題ありません。
- 副問合せでは、2種類の比較条件（単一行演算子と複数行演算子）が使用されます。

副問合せの種類

- 単一行副問合せ



- 複数行副問合せ



ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せの種類

- 単一行副問合せ: 内側のSELECT文から1行のみを戻す問合せ
- 複数行副問合せ: 内側のSELECT文から複数行を戻す問合せ

注意: これらの他に複数列副問合せもあります。これは、内側のSELECT文から複数の列を戻す問合せです。複数列副問合せについては、『Oracle Database 11g: 入門 SQL 基礎 II Ed 1』コースで説明します。

章の講義項目

- 副問合せ: 種類、構文およびガイドライン
- 単一行副問合せ
 - 副問合せにおけるグループ関数
 - HAVING句での副問合せ
- 複数行副問合せ
 - ALL演算子またはANY演算子の使用
- 副問合せにおけるNULL値

ORACLE

Copyright © 2007, Oracle. All rights reserved.

単一行副問合せ

- 1行のみを戻します。
- 単一行比較演算子を使用します。

演算子	意味
=	等しい
>	大きい
>=	以上
<	小さい
<=	以下
<>	等しくない

ORACLE

Copyright © 2007, Oracle. All rights reserved.

単一行副問合せ

単一行副問合せは、内側のSELECT文から1行のみを戻す問合せです。この種類の問合せでは、単一行演算子が使用されます。スライドには、単一行演算子の一覧が示されています。

例

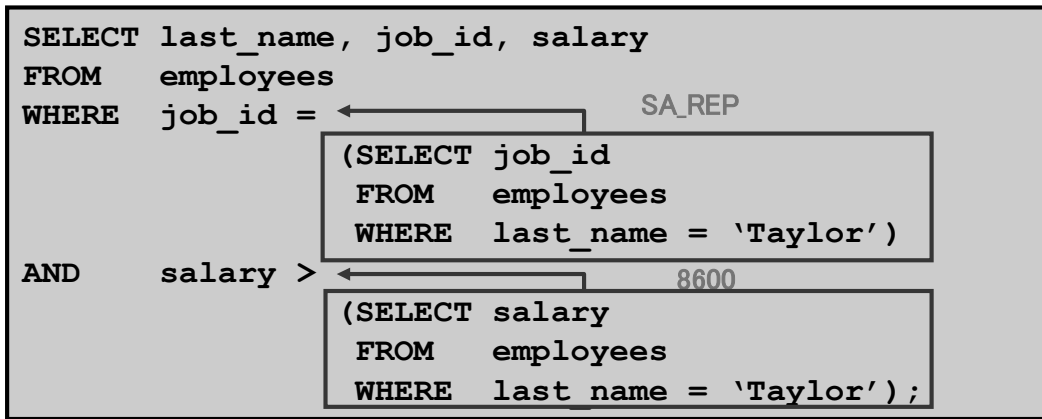
従業員141と同じ職務IDを持つ従業員を表示します。

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  employee_id = 141);
```

	LAST_NAME	JOB_ID
1	Rajs	ST_CLERK
2	Davies	ST_CLERK
3	Matos	ST_CLERK
4	Vargas	ST_CLERK

単一行副問合せの実行

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = (SELECT job_id
                  FROM   employees
                  WHERE  last_name = 'Taylor')
AND    salary > (SELECT salary
                  FROM   employees
                  WHERE  last_name = 'Taylor');
```



R	LAST_NAME	R	JOB_ID	R	SALARY
1	Abel		SA_REP		11000

ORACLE

Copyright © 2007, Oracle. All rights reserved.

単一行副問合せの実行

SELECT文は問合せブロックと考えることができます。スライドの例では、Taylorと職務が同じで、Taylorより給与の多い従業員が表示されます。

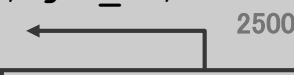
この例は、外側の問合せと2つの内側の問合せという3つの問合せブロックで構成されています。内側の問合せが最初に実行され、問合せ結果としてSA_REPおよび8600がそれぞれ取得されます。次に、外側の問合せブロックが処理され、内側の問合せから戻されたこれらの値を使用して検索条件が完成されます。

両方の内側の問合せからそれぞれ単一の値(SA_REPおよび8600)が戻されるため、このSQL文は単一行副問合せと呼ばれます。

注意: 外側の問合せと内側の問合せは、異なる表からデータを取得することも可能です。

副問合せにおけるグループ関数の使用方法

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
      (SELECT MIN(salary)
       FROM   employees);
```



	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せにおけるグループ関数の使用方法

副問合せでグループ関数を使用して単一行を戻し、それを基に主問合せのデータを表示することができます。副問合せはカッコで囲み、比較条件の後に配置します。

スライドの例では、給与が最低給与に等しいすべての従業員の姓、職務IDおよび給与が表示されます。MINグループ関数は、外側の問合せに単一の値(2500)を戻します。

HAVING句での副問合せ

- Oracleサーバーは、副問合せを最初に実行します。
- Oracleサーバーは、主問合せのHAVING句に結果を戻します。

```
SELECT  department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING  MIN(salary) > (SELECT MIN(salary)
                       FROM    employees
                       WHERE   department_id = 50);
```

2500

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	90	17000
3	20	6000
...		
7	10	4400

ORACLE

Copyright © 2007, Oracle. All rights reserved.

HAVING句での副問合せ

副問合せは、WHERE句のみではなく、HAVING句でも使用できます。Oracleサーバーで副問合せが実行されると、その結果は主問合せのHAVING句に戻されます。

スライドのSQL文では、最低給与が部門50の最低給与より高い部門がすべて表示されます。

例

平均給与が最も低い職務を検索します。

```
SELECT  job_id, AVG(salary)
FROM    employees
GROUP BY job_id
HAVING  AVG(salary) = (SELECT  MIN(AVG(salary))
                       FROM    employees
                       GROUP BY job_id);
```

	JOB_ID	AVG(SALARY)
1	ST_CLERK	2925

この文の間違いは何か

```
SELECT employee_id, last_name
FROM employees
WHERE salary =
  (SELECT MIN(salary)
   FROM employees
   GROUP BY department_id);
```

ORA-01427: single-row subquery returns more than one ...



An error was encountered performing the requested operation:

ORA-01427: single-row subquery returns more than one row
01427.00000 - "single-row subquery returns more than one row"

*Cause:

*Action:

Error at Line:1

複数行副問合せでの
単一行演算子の使用

ORACLE

Copyright © 2007, Oracle. All rights reserved.

この文の間違いは何か

副問合せで頻繁に発生するエラーは、単一行副問合せに対して複数行が戻されるというものです。

スライドのSQL文では、副問合せにGROUP BY句が含まれるため、この副問合せからは、見つかったグループごとに1行ずつ、複数の行が戻されることになります。この例の場合、副問合せの結果は4400、6000、2500、4200、7000、17000および8300です。

外側の問合せはこれらの結果を受け取り、それをWHERE句で使用します。WHERE句にある等号(=)演算子は、単一の値を前提にした単一行比較演算子です。等号演算子では、副問合せからの複数の値を受け付けることができないため、エラーが発生します。

このエラーを修正するには、等号(=)演算子をINに変更します。

内側の問合せから行が戻されない

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  last_name = 'Haas');
```

0 rows selected

“Haas”という名前の従業員は存在しないため、
副問合せから行が戻されません。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

内側の問合せから行が戻されない

副問合せで頻繁に発生する問題は、内側の問合せから行が戻されないというものです。

スライドのSQL文では、副問合せにWHERE句が含まれます。この文の意図はHaasという名前の従業員を探すことであると推測されます。この文は正しいのですが、実行しても行は選択されません。

これは、Haasという名前の従業員が存在しないためです。したがって、副問合せからは行が戻されません。外側の問合せは、副問合せの結果(NULL)を受け取り、WHERE句で使用します。職務IDがNULLの従業員は見つからない、外側の問合せからも行は戻されません。NULLの値を持つ職務が存在する場合でも、その行は戻されません。2つのNULL値の比較の結果はNULLになり、WHERE条件は真にならないためです。

章の講義項目

- 副問合せ: 種類、構文およびガイドライン
- 単一行副問合せ
 - 副問合せにおけるグループ関数
 - HAVING句での副問合せ
- 複数行副問合せ
 - ALL演算子またはANY演算子の使用
- 副問合せにおけるNULL値

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数行副問合せ

- 複数の行を戻します。
- 複数行比較演算子を使用します。

演算子	意味
IN	リスト内のいずれかのメンバーと等しい。
ANY	直前に=、!=、>、<、<=、>=が必要。1つの値を、リスト内の値または問合せで戻されるそれぞれの値と比較。問合せから行が戻されない場合、FALSEと評価される。
ALL	直前に=、!=、>、<、<=、>=が必要。1つの値を、リスト内の値または問合せで戻される値のすべてと比較。問合せから行が戻されない場合、TRUEと評価される。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数行副問合せ

複数行を戻す副問合せは、複数行副問合せと呼ばれます。複数行副問合せでは、単数行演算子ではなく、複数行演算子を使用します。複数行演算子は、次のように1つ以上の値を想定しています。

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (SELECT MIN(salary)
                  FROM employees
                  GROUP BY department_id);
```

例

給与が各部門の最低給与と等しい従業員を調べます。

内側の問合せが最初に実行され、問合せ結果が出力されます。次に主問合せブロックが処理され、内側の問合せから戻された値を使用して検索条件を完成させます。この主問合せは、実際にはOracleサーバーから次のように見えます。

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300,
                 8600, 17000);
```

複数行副問合せでのANY演算子の使用方法

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数行副問合せでのANY演算子の使用方法

ANY演算子(およびそのシノニムであるSOME演算子)は、1つの値を、副問合せから戻されるそれぞれの値と比較します。スライドの例では、ITプログラマーではなく、給与がいずれかのITプログラマーの給与より少ない従業員が表示されます。プログラマーの最高給与は\$9,000です。

<ANYは最大値より小さいことを意味します。>ANYは最小値より大きいことを意味します。=ANYはINと同等です。

複数行副問合せでのALL演算子の使用方法

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

ORACLE

Copyright © 2007, Oracle. All rights reserved.

複数行副問合せでのALL演算子の使用方法

ALL演算子は、1つの値を、副問合せから戻されるすべての値と比較します。スライドの例では、職務IDがIT_PROGであるすべての従業員の給与より給与が少なく、職務がIT_PROGではない従業員が表示されます。

>ALLは最大値より大きいことを意味し、<ALLは最小値より小さいことを意味します。

IN、ANYおよびALL演算子は、NOT演算子と併用できます。

章の講義項目

- 副問合せ: 種類、構文およびガイドライン
- 単一行副問合せ
 - 副問合せにおけるグループ関数
 - HAVING句での副問合せ
- 複数行副問合せ
 - ALL演算子またはANY演算子の使用
- 副問合せにおけるNULL値

ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せにおけるNULL値

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
                                (SELECT mgr.manager_id
                                FROM   employees mgr);
```

0 rows selected

ORACLE

Copyright © 2007, Oracle. All rights reserved.

副問合せにおけるNULL値

スライドのSQL文は、部下を持たない従業員をすべて表示しようと試みたものです。理論的には、このSQL文は12行を戻すはずだったのですが、1行も戻していません。内側の問合せから戻される値の1つがNULL値であるため、問合せ全体としては行が戻されません。

これは、NULL値と比較する条件はすべてNULLになるためです。したがって、副問合せの結果セットの一部にNULL値が含まれる可能性がある場合には、NOT IN演算子を使用しないようにします。NOT IN演算子は<> ALLと同等です。

IN演算子を副問合せで使用する場合には、結果セットにNULL値が含まれていても問題はありません。IN演算子は=ANYと同等です。たとえば、部下を持つ従業員を表示するには、次のSQL文を使用します。

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id IN
                                (SELECT mgr.manager_id
                                FROM   employees mgr);
```

副問合せにおけるNULL値(続き)

別の方法として、副問合せでWHERE句を使用して、部下を持たない従業員をすべて表示することもできます。

```
SELECT last_name FROM employees
WHERE  employee_id NOT IN
        (SELECT manager_id
         FROM   employees
         WHERE  manager_id IS NOT NULL);
```

Oracle Internal & Oracle Academy
Use Only

まとめ

この章では、次のことを学習しました。

- 副問合せが問題の解決に役立つ状況の理解
- 問合せが未知の値に基づく場合の副問合せの記述方法

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT select_list
         FROM    table);
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

この章では、副問合せの使用方法について学習しました。副問合せとは、別のSQL文の句内に埋め込まれたSELECT文のことです。副問合せは、未知の中間値を使用する検索条件に基づいて問合せを実行する場合に便利です。

副問合せの特性は、次のとおりです。

- 単一行演算子(=、<>、>、>=、<、<=など)を含む主文に1行のデータを渡すことができます。
- 複数行演算子(INなど)を含む主文に複数行のデータを渡すことができます。
- Oracleサーバーで最初に処理され、その後、その結果がWHERE句やHAVING句で使用されます。
- グループ関数を含むことができます。

演習7: 概要

この演習では次の項目について説明しています。

- 未知の基準に基づいて値を問い合わせのための副問合せの作成方法
- あるデータセットに存在し、別のデータセットには存在しない値を検索するための副問合せの使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

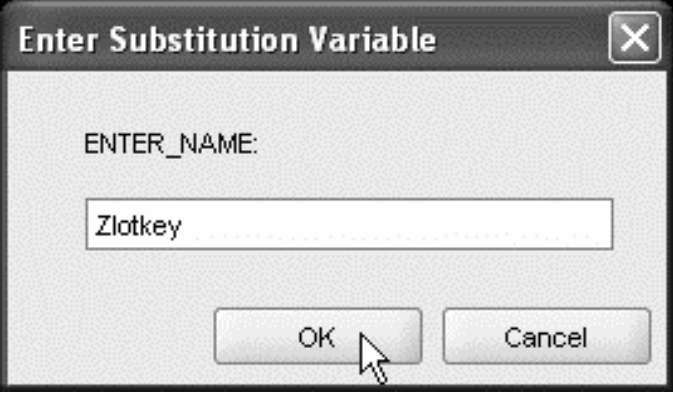
演習7: 概要

この演習では、ネストしたSELECT文を使用した複雑な問合せを記述します。

演習問題においては、内側の問合せを先に作成する方がよいでしょう。外側の問合せのコードを記述する前に、内側の問合せを実行して、予想通りのデータが取得されることを確認してください。


演習7

1. HR部門は、ユーザーに従業員の姓の入力を求めるプロンプトを表示し、入力された名前の従業員と同じ部門に所属するすべての従業員（入力した従業員は除く）の姓と雇用日を表示する問合せを必要としています。たとえば、ユーザーがZlotkeyと入力すると、Zlotkeyとともに勤務するすべての従業員（Zlotkeyは除く）が検索されます。



	 LAST_NAME	HIRE_DATE
1	Abel	11-MAY-96
2	Taylor	24-MAR-98

2. 給与が平均給与より多いすべての従業員の従業員番号、姓および給与を表示するレポートを作成してください。給与の昇順に結果をソートしてください。

	 EMPLOYEE_ID	 LAST_NAME	 SALARY
1	103	Hunold	9000
2	149	Zlotkey	10500
3	174	Abel	11000
4	205	Higgins	12000
5	201	Hartstein	13000
6	101	Kochhar	17000
7	102	De Haan	17000
8	100	King	24000

演習7(続き)

3. 姓に“u”の文字が含まれる従業員が所属する部門に勤務するすべての従業員の従業員番号と姓を表示する問合せを記述します。作成したSQL文は、lab_07_03.sqlとして保存します。この問合せを実行してください。

	EMPLOYEE_ID	LAST_NAME
1	124	Mourgos
2	141	Rajs
3	142	Davies
4	143	Matos
5	144	Vargas
6	103	Hunold
7	104	Ernst
8	107	Lorentz

4. HR部門では、部門の所在地IDが1700であるすべての従業員の姓、部門番号および職務IDを表示するレポートを必要としています。

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Whalen	10	AD_ASST
2	King	90	AD_PRES
3	Kochhar	90	AD_VP
4	De Haan	90	AD_VP
5	Higgins	110	AC_MGR
6	Gietz	110	AC_ACCOUNT




問合せを変更して、ユーザーに所在地IDの入力を求めるプロンプトが表示されるようにします。この問合せは、lab_07_04.sqlという名前のファイルに保存します。

5. HR用に、Kingを上司とするすべての従業員の姓と給与を表示するレポートを作成します。

	LAST_NAME	SALARY
1	Kochhar	17000
2	De Haan	17000
3	Mourgos	5800
4	Zlotkey	10500
5	Hartstein	13000

演習7(続き)

6. HR用に、Executive部門に所属するすべての従業員の部門番号、姓および職務IDを表示するレポートを作成します。

	 DEPARTMENT_ID	 LAST_NAME	 JOB_ID
1	90	King	AD_PRES
2	90	Kochhar	AD_VP
3	90	De Haan	AD_VP

時間があるときは、次の演習問題に進みます。

7. lab_07_03.sqlの問合せを変更し、給与が平均給与より多い従業員で、姓に“u”の文字が含まれる従業員が所属する部門に勤務するすべての従業員の従業員番号、姓および給与が表示されるようにしてください。lab_07_03.sqlをlab_07_07.sqlとして保存し直します。lab_07_07.sqlの文を実行してください。

	 EMPLOYEE_ID	 LAST_NAME	 SALARY
1	103	Hunold	9000

Oracle Internal & Oracle Academy
Use Only

Oracle Internal & Oracle Academy
Use Only

8 集合演算子の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracle Internal & Oracle Academy
Use Only

目的

この章を終えると、次のことができるようになります。

- 集合演算子の説明
- 集合演算子を使用して、複数の問合せを単一の問合せに組み合わせる
- 戻される行の順序を制御する

ORACLE

Copyright © 2007, Oracle. All rights reserved.

目的

この章では、集合演算子を使用して問合せを記述する方法について学習します。

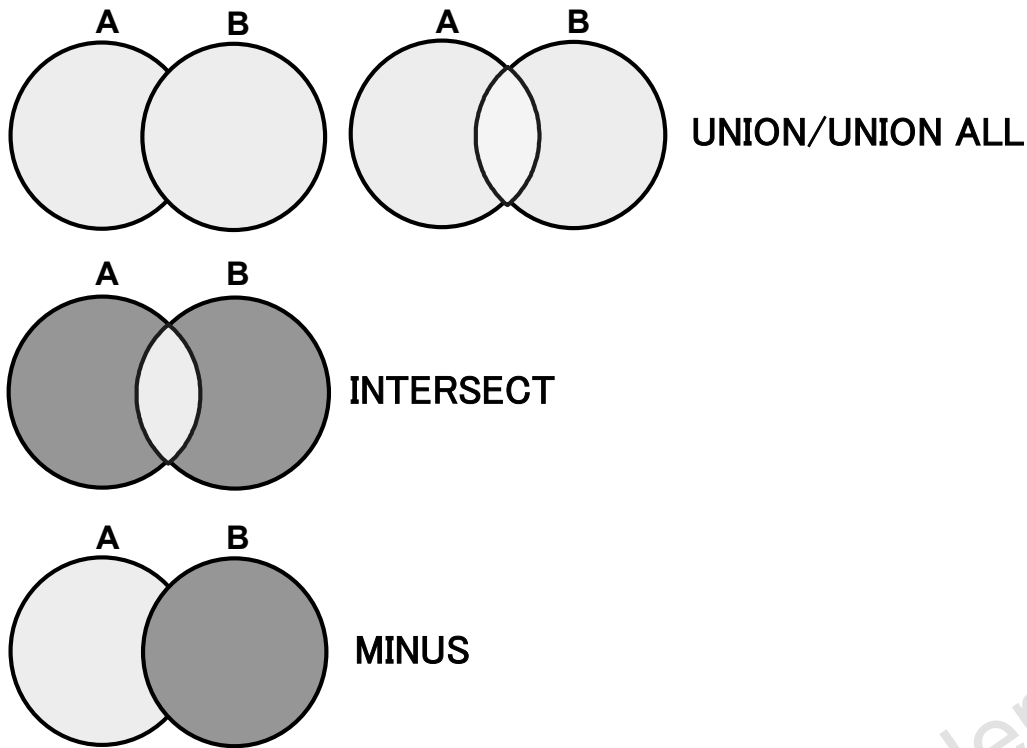
章の講義項目

- 集合演算子: 種類およびガイドライン
- この章で使用する表
- UNION演算子とUNION ALL演算子
- INTERSECT演算子
- MINUS演算子
- SELECT文の一致
- 集合演算子でのORDER BY句の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

集合演算子



ORACLE

Copyright © 2007, Oracle. All rights reserved.

集合演算子

集合演算子は、複数のコンポーネント問合せの結果を1つの結果にまとめます。集合演算子を含む問合せを複合問合せと呼びます。

演算子	戻り値
UNION	重複を除去した後の、両方の問合せからの行
UNION ALL	すべての重複を含む、両方の問合せからの行
INTERSECT	両方の問合せに共通の行
MINUS	最初の問合せには存在し、2番目の問合せには存在しない行

集合演算子の優先順位はすべて同じです。SQL文に複数の集合演算子が含まれる場合、Oracleサーバーは左(上)から右(下)の順に評価します(カッコを使用して別の順序を明示的に指定していない場合)。INTERSECT演算子を他の集合演算子とともに使用する問合せにおいては、カッコを使用して評価順序を明示的に指定する必要があります。

集合演算子のガイドライン

- SELECTリスト内の式の数が一致している必要があります。
- 2番目の問合せの各列のデータ型は、最初の問合せの対応する列のデータ型と一致する必要があります。
- カッコを使用して実行順序を変更することができます。
- ORDER BY句は、文の最後にしか使用できません。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

集合演算子のガイドライン

- 問合せのSELECTリスト内の式は、数とデータ型が一致する必要があります。WHERE句でUNION、UNION ALL、INTERSECTおよびMINUS演算子を使用する問合せは、SELECTリストと同じ列数およびデータ型を持つ必要があります。複合問合せ内の問合せにあるSELECTリストの列のデータ型は、まったく同じである必要はありません。2番目の問合せの列は、最初の問合せの対応する列と同じデータ型グループ(数値、文字など)であることが必要です。
- 集合演算子は、副問合せでも使用できます。
- INTERSECT演算子を他の集合演算子とともに使用する問合せにおいては、カッコを使用して評価順序を指定する必要があります。これにより、新たなSQL規格でINTERSECT演算子に他の集合演算子より高い優先順位が与えられた場合にも、その規格への準拠が保証されます。

Oracleサーバーと集合演算子

- UNION ALL以外では、重複する行は自動的に除去されます。
- 最初の問合せの列名が結果に表示されます。
- UNION ALL以外では、デフォルトで出力が昇順にソートされます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

Oracleサーバーと集合演算子

問合せに集合演算子を使用すると、UNION ALL演算子の場合を除き、Oracleサーバーでは重複する行が自動的に除去されます。出力の列名は、最初のSELECT文の列リストによって決定されます。デフォルトでは、SELECT句の最初の列を基準に昇順に出力がソートされます。

複合問合せのコンポーネントである各問合せのSELECTリストの対応する式は、数およびデータ型が一致する必要があります。コンポーネント問合せで文字データが選択される場合、戻り値のデータ型は次のように決定されます。

- 両方の問合せで同じ長さのCHARデータ型の値が選択される場合、戻り値はその長さのCHARデータ型になります。問合せで異なる長さのCHARデータ型の値が選択される場合、戻り値は、長い方のCHARデータ値の長さを持つVARCHAR2データ型になります。
- 問合せのいずれか、または両方でVARCHAR2データ型の値が選択される場合、戻り値はVARCHAR2データ型になります。

コンポーネント問合せで数値データが選択される場合、戻り値のデータ型は数値の優先順位で決定されます。すべての問合せでNUMBER型の値が選択される場合、戻り値はNUMBERデータ型になります。集合演算子を使用する問合せの場合、Oracleサーバーではデータ型のグループを超えた変換は暗黙的には実行されません。したがって、各コンポーネント問合せの対応する式がそれぞれ文字データと数値データになる場合、Oracleサーバーからエラーが戻されます。

章の講義項目

- 集合演算子: 種類およびガイドライン
- この章で使用する表
- UNION演算子とUNION ALL演算子
- INTERSECT演算子
- MINUS演算子
- SELECT文の一致
- 集合演算子でのORDER BY句の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

この章で使用する表

この章で使用する表は次のとおりです。

- EMPLOYEES: 現行の全従業員に関する詳細データ
- JOB_HISTORY: 職務変更時の前職務の開始日と終了日、職務ID番号および部門に関する詳細データの記録

ORACLE

Copyright © 2007, Oracle. All rights reserved.

この章で使用する表

この章では、2つの表を使用します。EMPLOYEES表とJOB_HISTORY表です。

EMPLOYEES表はすでに使用している表です。この表には一意の識別番号、電子メールアドレス、職務ID (ST_CLERK、SA_REPなど)、給与、マネージャなど、従業員の詳細データが保存されています。

従業員によっては、この会社に長期間勤め、異なる職務に異動した経歴を持ちます。このような職務経歴は、JOB_HISTORY表に記録されています。従業員が職務を異動した場合、前職務の開始日と終了日、job_id (ST_CLERK、SA_REPなど) および部門がJOB_HISTORY表に記録されます。

EMPLOYEES表とJOB_HISTORY表の構造とデータについては、次のページに示します。

この章で使用する表(続き)

会社には、その会社の在職期間中に同じ職務に複数回就いたことのある従業員も含まれます。たとえば、1998年3月24日に雇用されたTaylorという従業員の場合、1998年3月24日～1998年12月31日の期間は職種SA_REPに就き、1999年1月1日から1999年12月31日の期間は職務SA_MANに就きました。その後、職務SA_REPに戻り、これが現行の職務です。

DESCRIBE employees

DESCRIBE employees		
Name	Null	Type
-----	-----	-----
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

この章で使用する表(続き)

```
SELECT employee_id, last_name, job_id, hire_date, department_id
FROM employees;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
1	100	King	AD_PRES	17-JUN-87	90
2	101	Kochhar	AD_VP	21-SEP-89	90
3	102	De Haan	AD_VP	13-JAN-93	90
4	103	Hunold	IT_PROG	03-JAN-90	60
5	104	Ernst	IT_PROG	21-MAY-91	60
6	107	Lorentz	IT_PROG	07-FEB-99	60
7	124	Mourgos	ST_MAN	16-NOV-99	50
8	141	Rajs	ST_CLERK	17-OCT-95	50
9	142	Davies	ST_CLERK	29-JAN-97	50
10	143	Matos	ST_CLERK	15-MAR-98	50
11	144	Vargas	ST_CLERK	09-JUL-98	50
12	149	Zlotkey	SA_MAN	29-JAN-00	80
13	174	Abel	SA_REP	11-MAY-96	80
14	176	Taylor	SA_REP	24-MAR-98	80
15	178	Grant	SA_REP	24-MAY-99	(null)
16	200	Whalen	AD_ASST	17-SEP-87	10
17	201	Hartstein	MK_MAN	17-FEB-96	20

...

```
DESCRIBE job_history
```

describe job_history		
Name	Null	Type
-----	-----	-----
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

この章で使用する表(続き)

```
SELECT * FROM job_history;
```

	<small>AZ</small> EMPLOYEE_ID	START_DATE	END_DATE	<small>AZ</small> JOB_ID	<small>AZ</small> DEPARTMENT_ID
1	102	13-JAN-93	24-JUL-98	IT_PROG	60
2	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
3	101	28-OCT-93	15-MAR-97	AC_MGR	110
4	201	17-FEB-96	19-DEC-99	MK_REP	20
5	114	24-MAR-98	31-DEC-99	ST_CLERK	50
6	122	01-JAN-99	31-DEC-99	ST_CLERK	50
7	200	17-SEP-87	17-JUN-93	AD_ASST	90
8	176	24-MAR-98	31-DEC-98	SA_REP	80
9	176	01-JAN-99	31-DEC-99	SA_MAN	80
10	200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

Oracle Internal & Oracle Academy
Use Only

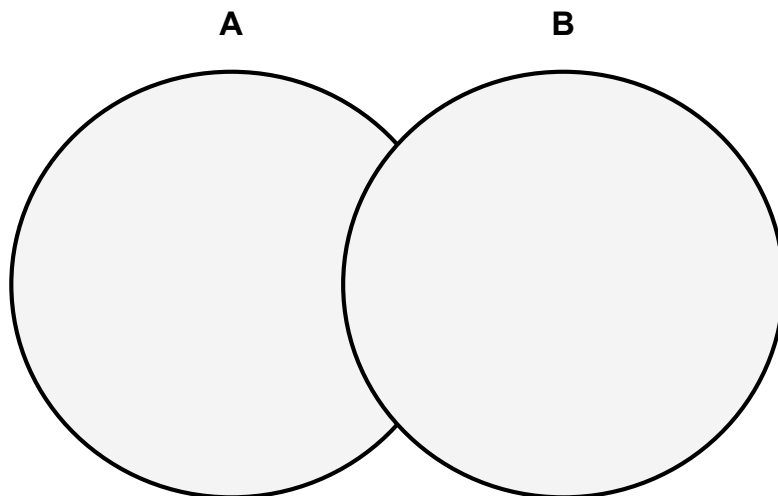
章の講義項目

- 集合演算子: 種類およびガイドライン
- この章で使用する表
- **UNION演算子とUNION ALL演算子**
- INTERSECT演算子
- MINUS演算子
- SELECT文の一致
- 集合演算子でのORDER BY句の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

UNION演算子



UNION演算子は、重複を除去した後の両方の問合せからの行を戻します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

UNION演算子

UNION演算子は、どちらかの問合せで選択された行をすべて戻します。複数の表からすべての行を戻し、重複する行を除去する場合にUNION演算子を使用します。

ガイドライン

- 選択される列数は同じである必要があります。
- 選択される列のデータ型は、同じデータ型グループ(数値、文字など)である必要があります。
- 列の名前は同一である必要はありません。
- UNION演算子は、選択された列のすべてに対して作用します。
- 重複チェックの際、NULL値は無視されません。
- デフォルトで、SELECT句の列を基準に昇順に出力がソートされます。

UNION演算子の使用方法

すべての従業員の現行の職務と以前の職務の詳細を表示します。
各従業員は一度だけ表示されます。

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID
1	100 AD_PRES
2	101 AC_ACCOUNT
...	
22	200 AC_ACCOUNT
23	200 AD_ASST
24	201 MK_MAN
...	

ORACLE

Copyright © 2007, Oracle. All rights reserved.

UNION演算子の使用方法

UNION演算子では、重複するレコードがすべて除去されます。EMPLOYEES表とJOB_HISTORY表の両方で出現するレコードが同一の場合、これらのレコードは一度だけ表示されます。スライドに示された出力ではEMPLOYEE_IDが200の従業員レコードが2回表示されていますが、この理由はJOB_IDがそれぞれの行で異なるためです。

次の例を検討してみます。

```
SELECT employee_id, job_id, department_id
FROM   employees
UNION
SELECT employee_id, job_id, department_id
FROM   job_history;
```

UNION演算子の使用方法(続き)

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
2	101	AC_ACCOUNT	110

...

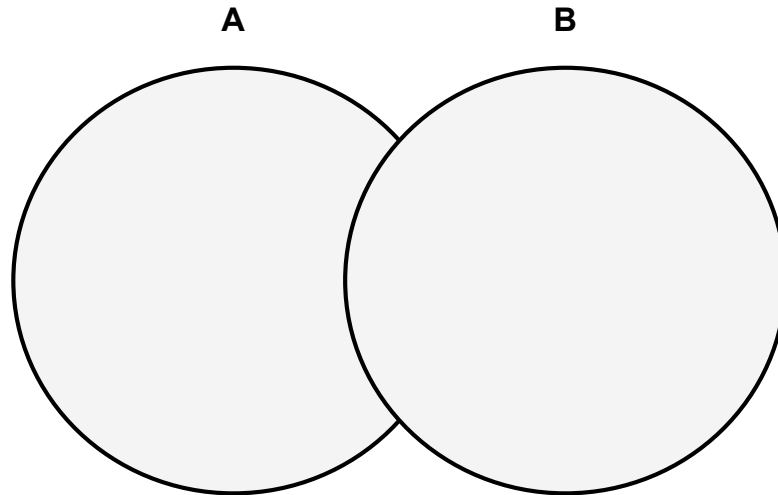
22	200	AC_ACCOUNT	90
23	200	AD_ASST	10
24	200	AD_ASST	90

...

この出力においては、従業員200が3回表示されます。この理由は、従業員200のDEPARTMENT_IDの値にあります。1つ目の行ではDEPARTMENT_IDが90、2つ目の行では10、そして3つ目は90です。職務IDと部門IDのこれらの組合せは一意であるため、従業員200の各行は一意であり、重複とはみなされません。SELECT句の最初の列(この場合はEMPLOYEE_ID)の昇順に出力がソートされている点にも注目してください。

Oracle Internal & Oracle Academy
Use Only

UNION ALL演算子



UNION ALL演算子は、すべての重複を含めて、両方の問合せからの行を戻します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

UNION ALL演算子

UNION ALL演算子は、複数の問合せからすべての行を戻す場合に使用します。

ガイドライン

UNION ALLの場合、重複する行は除去されず、デフォルトでは出力はソートされない点が UNIONと異なりますが、それ以外についてはUNIONとUNION ALLのガイドラインは同じです。

NION ALL演算子の使用方法

すべての従業員の現行の部門と以前の部門を表示します。

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100 AD_PRES	90
...		
16	144 ST_CLERK	50
17	149 SA_MAN	80
18	174 SA_REP	80
19	176 SA_REP	80
20	176 SA_MAN	80
21	176 SA_REP	80
22	178 SA_REP	(null)
...		
30	206 AC_ACCOUNT	110

ORACLE

Copyright © 2007, Oracle. All rights reserved.

UNION ALL演算子の使用方法

この例では、30行が選択されています。2つの表の組合せの合計が30行になります。UNION ALL演算子では、重複する行は除去されません。UNIONは、どちらかの問合せで選択された重複しない行をすべて戻します。一方UNION ALLは、すべての重複を含めて、どちらかの問合せで選択された行をすべて戻します。スライドの問合せを検討してみます。これはUNION句で記述されています。

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

この問合せでは、29行が戻されます。その理由は、重複している次の行が除去されるためです。

176 SA_REP	80
------------	----

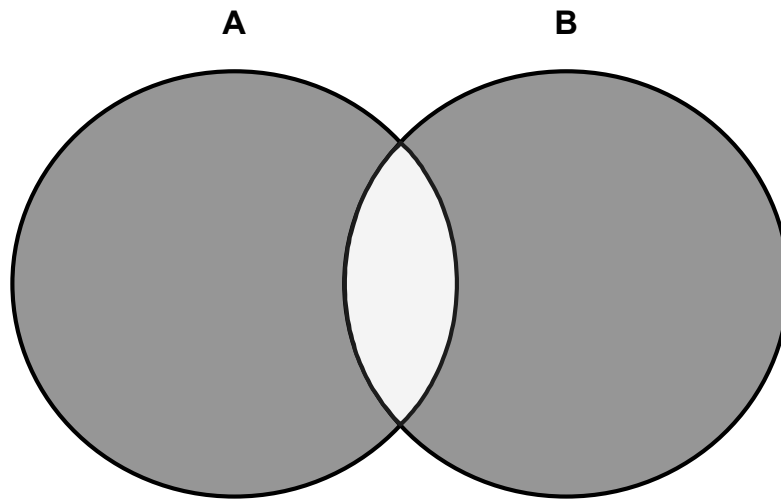
章の講義項目

- 集合演算子: 種類およびガイドライン
- この章で使用する表
- UNION演算子とUNION ALL演算子
- **INTERSECT演算子**
- MINUS演算子
- SELECT文の一致
- 集合演算子でのORDER BY句の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

INTERSECT演算子



INTERSECT演算子は、両方の問合せに共通の行を戻します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

INTERSECT演算子

INTERSECT演算子は、複数の問合せに共通の行をすべて戻す場合に使用します。

ガイドライン

- 問合せのSELECT文で選択される列の数とデータ型は、その問合せで使用するすべてのSELECT文において同一である必要があります。ただし、列の名前は同一である必要はありません。
- 交差する表の順序を逆にしても、結果は変わりません。
- INTERSECTではNULL値は無視されません。

INTERSECT演算子の使用方法

現行の職務と以前の職務が同じ(つまり、職務を異動したが、現在は以前と同じ職務に戻っている)従業員の従業員IDと職務IDを表示します。

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

ORACLE

Copyright © 2007, Oracle. All rights reserved.

INTERSECT演算子の使用方法

スライドの問合せの例では、両方の表で選択された列の値が一致するレコードのみが戻されます。EMPLOYEES表に対するSELECT文にDEPARTMENT_ID列を追加し、JOB_HISTORY表に対するSELECT文にDEPARTMENT_ID列を追加して問合せを実行した場合、結果はどうなるでしょうか? 別の列の追加により値の重複判定が変化する可能性があるため、結果が異なる可能性があります。

例

```
SELECT employee_id, job_id, department_id
FROM employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM job_history;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	176	SA_REP	80

EMPLOYEES.DEPARTMENT_IDの値がJOB_HISTORY.DEPARTMENT_IDの値と異なるため、従業員200は結果には含まれなくなります。

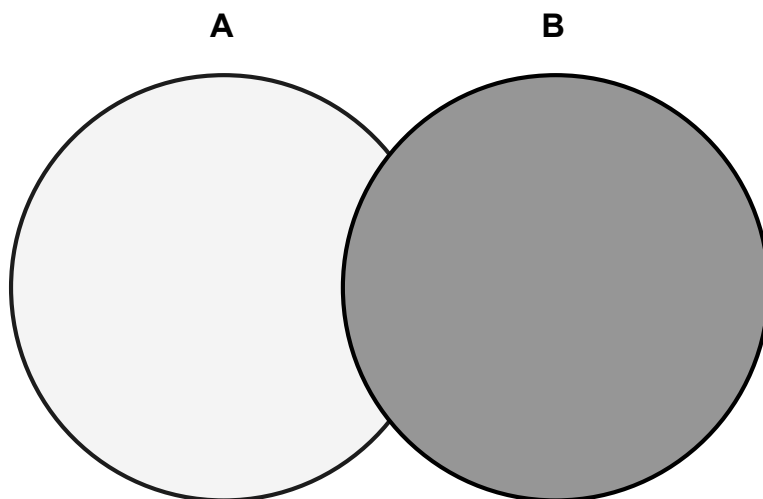
章の講義項目

- 集合演算子: 種類およびガイドライン
- この章で使用する表
- UNION演算子とUNION ALL演算子
- INTERSECT演算子
- **MINUS演算子**
- SELECT文の一致
- 集合演算子でのORDER BY句の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

MINUS演算子



MINUS演算子は、最初の間合せで選択された行で、2番目の間合せの結果セットには存在しない行(重複しない行)をすべて戻します。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

MINUS演算子

MINUS演算子は、最初の間合せで選択された行で、2番目の間合せの結果セットには存在しない行(重複しない行)をすべて戻す場合に使用します。つまり、最初のSELECT文から2番目のSELECT文を引き(MINUS)ます。

注意: 間合せで使用するすべてのSELECT文において、間合せのSELECT文で選択される列の数が同じで、データ型が同じデータ型グループに属する必要があります。ただし、列の名前は同一である必要はありません。

MINUS演算子の使用方法

職務を一度も異動したことがない従業員の従業員IDを表示します。

```
SELECT employee_id
FROM   employees
MINUS
SELECT employee_id
FROM   job_history;
```

	EMPLOYEE_ID
1	100
2	103
3	104
4	107
5	124
...	
14	205
15	206

ORACLE

Copyright © 2007, Oracle. All rights reserved.

MINUS演算子の使用方法

スライドの例では、EMPLOYEES表の従業員IDからJOB_HISTORY表の従業員IDが除去されます。結果セットには、この除去処理後の残りの従業員が表示されます。つまり、これらの従業員は、EMPLOYEES表に存在し、JOB_HISTORY表には存在しない行によって表されます。これらの行は、一度も職務を異動したことがない従業員のレコードです。

章の講義項目

- 集合演算子: 種類およびガイドライン
- この章で使用する表
- UNION演算子とUNION ALL演算子
- INTERSECT演算子
- MINUS演算子
- SELECT文の一致
- 集合演算子でのORDER BY句の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SELECT文の一致

- UNION演算子を使用して、所在地ID、部門名および所在地の州を表示します。
- どちらか一方の表に列が存在しない場合、データ型を一致させる操作が必要になります (TO_CHAR関数または他の変換関数を使用)。

```
SELECT location_id, department_name "Department",  
       TO_CHAR(NULL) "Warehouse location"  
FROM departments  
UNION  
SELECT location_id, TO_CHAR(NULL) "Department",  
       state_province  
FROM locations;
```

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SELECT文の一致

問合せのSELECTリストに記述する式は数が一致する必要があり、この規則に準拠するために、ダミー列とデータ型変換関数を使用することができます。スライドでは、Warehouse locationという名前がダミーの列ヘッダーとして指定されています。また、2番目の問合せから取得されるstate_province列のVARCHAR2データ型と一致させるために、最初の問合せでTO_CHAR関数が使用されています。同様に、最初の問合せから取得されるdepartment_name列のVARCHAR2データ型と一致させるために、2番目の問合せでTO_CHAR関数が使用されています。

この問合せの出力を次に示します。

	LOCATION_ID	Department	Warehouse location
1	1400	IT	(null)
2	1400	(null)	Texas
3	1500	Shipping	(null)
4	1500	(null)	California
5	1700	Accounting	(null)
6	1700	Administration	(null)
7	1700	Contracting	(null)
8	1700	Executive	(null)
9	1700	(null)	Washington

...

SELECT文の一致: 例

UNION演算子を使用して、全従業員の従業員ID、職務IDおよび給与を表示します。

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID	SALARY
1	100	AD_PRES	24000
2	101	AC_ACCOUNT	0
3	101	AC_MGR	0
4	101	AD_VP	17000
5	102	AD_VP	17000
...			
29	205	AC_MGR	12000
30	206	AC_ACCOUNT	8300

ORACLE

Copyright © 2007, Oracle. All rights reserved.

SELECT文の一致: 例

EMPLOYEES表とJOB_HISTORY表には、共通の列がいくつかあります (EMPLOYEE_ID、JOB_ID、DEPARTMENT_IDなど)。給与はEMPLOYEES表のみに存在するのですが、問合せでUNION演算子を使用して、従業員ID、職務IDおよび給与を表示する場合を考えてみます。

スライドのコード例では、EMPLOYEES表とJOB_HISTORY表のEMPLOYEE_ID列およびJOB_ID列は一致しています。EMPLOYEESのSELECT文のSALARY列は数値であるため、これに一致させるため、リテラル値の0がJOB_HISTORYのSELECT文に追加されています。

スライドに示された結果では、出力内で、JOB_HISTORY表のレコードに対応する各行のSALARY列には、0が表示されています。

章の講義項目

- 集合演算子: 種類およびガイドライン
- この章で使用する表
- UNION演算子とUNION ALL演算子
- INTERSECT演算子
- MINUS演算子
- SELECT文の一致
- 集合演算子でのORDER BY句の使用方法

ORACLE

Copyright © 2007, Oracle. All rights reserved.

集合演算子での ORDER BY句の使用方法

- ORDER BY句は、複合問合せの最後に一度しか使用できません。
- 各コンポーネント問合せで、個別にORDER BY句を使用することはできません。
- ORDER BY句では、最初のSELECT問合せの列のみが認識されます。
- デフォルトでは、最初のSELECT問合せの最初の列を使用して、出力が昇順にソートされます。

ORACLE

Copyright © 2007, Oracle. All rights reserved.

集合演算子でのORDER BY句の使用方法

ORDER BY句は、複合問合せでは一度しか使用できません。使用する場合、ORDER BY句は問合せの最後に配置する必要があります。ORDER BY句には、列名または別名を使用できます。デフォルトでは、最初のSELECT問合せの最初の列を基準に昇順に出力がソートされます。

注意: ORDER BY句では、2番目のSELECT問合せの列名は認識されません。列名の混乱を避けるため、通常はORDER BYに列の位置を指定します。

たとえば、次の文の出力は、job_idの昇順に表示されます。

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history
ORDER BY 2;
```

ORDER BYを省略した場合、つまりデフォルトでは、employee_idの昇順に出力がソートされます。2番目の問合せの列を使用して出力をソートすることはできません。

まとめ

この章では、次のことを学習しました。

- UNIONを使用して、すべての重複しない行を取得する方法
- UNION ALLを使用して、重複を含めたすべての行を取得する方法
- INTERSECTを使用して、両方の問合せに共通するすべての行を取得する方法
- MINUSを使用して、最初の問合せで選択され、2番目の問合せには存在しないすべての重複しない行を取得する方法
- ORDER BY句は文の最後でのみ使用すること

ORACLE

Copyright © 2007, Oracle. All rights reserved.

まとめ

- UNION演算子は、複合問合せの各問合せで選択された重複しない行をすべて戻します。複数の表からすべての行を戻し、重複する行は除去する場合にUNION演算子を使用します。
- UNION ALL演算子は、複数の問合せで選択されたすべての行を戻す場合に使用します。UNION演算子の場合と異なり、重複する行は除去されません。また、デフォルトでは出力はソートされません。
- INTERSECT演算子は、複数の問合せに共通の行をすべて戻す場合に使用します。
- MINUS演算子は、最初の問合せで戻された行で、2番目の問合せに存在しない行を戻す場合に使用します。
- ORDER BY句は、複合文の最後にしか使用できないことに注意が必要です。
- SELECTリストの対応する式は、数とデータ型が必ず一致している必要があります。

演習8: 概要

この演習では、次の演算子を使用してレポートを作成します。

- UNION演算子
- INTERSECTION演算子
- MINUS演算子

ORACLE

Copyright © 2007, Oracle. All rights reserved.

演習8: 概要

この演習では、集合演算子を使用した問合せを記述します。

演習8

1. HR部門では、職務IDのST_CLERKが含まれない部門の部門IDのリストを必要としています。集合演算子を使用してこのレポートを作成してください。

	DEPARTMENT_ID
1	10
2	20
3	60
4	80
5	90
6	110
7	190

2. HR部門では、部門が存在しない国のリストを必要としています。それらの国の国IDと国名を表示します。集合演算子を使用してこのレポートを作成してください。

COUNTRY_ID	COUNTRY_NAME
1 DE	Germany

3. 部門10、50および20に対する職務のリストをこの順に出力します。集合演算子を使用して職務IDと部門IDを表示してください。

	JOB_ID	DEPARTMENT_ID
1	AD_ASST	10
2	ST_MAN	50
3	ST_CLERK	50
4	MK_MAN	20
5	MK_REP	20

4. 現行の職務が会社に雇用された当初の職務と同じ(つまり、職務を異動したが、現在は元の職務に戻っている)従業員の従業員IDと職務IDのリストを表示するレポートを作成します。

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

演習8(続き)

5. HR部門では、次の仕様のレポートを必要としています。

- EMPLOYEES表のすべての従業員の姓と部門ID(部門に所属しているかどうかに関係なく)
- DEPARTMENTS表のすべての部門の部門IDと部門名(所属する従業員が存在するかどうかに関係なく)

複合問合せを記述して、この仕様を実現してください。

	A2	LAST_NAME	A2	DEPARTMENT_ID	A2	TO_CHAR(NULL)
1		Abel		80		(null)
2		Davies		50		(null)
3		De Haan		90		(null)
4		Ernst		60		(null)
5		Fay		20		(null)
6		Gietz		110		(null)
7		Grant		(null)		(null)
8		Hartstein		20		(null)
9		Higgins		110		(null)
10		Hunold		60		(null)
11		King		90		(null)
12		Kochhar		90		(null)
13		Lorentz		60		(null)
14		Matos		50		(null)
15		Mourgos		50		(null)
16		Rajs		50		(null)
17		Taylor		80		(null)
18		Vargas		50		(null)

19		Whalen		10		(null)
20		Zlotkey		80		(null)
21		(null)		10		Administration
22		(null)		20		Marketing
23		(null)		50		Shipping
24		(null)		60		IT
25		(null)		80		Sales
26		(null)		90		Executive
27		(null)		110		Accounting
28		(null)		190		Contracting